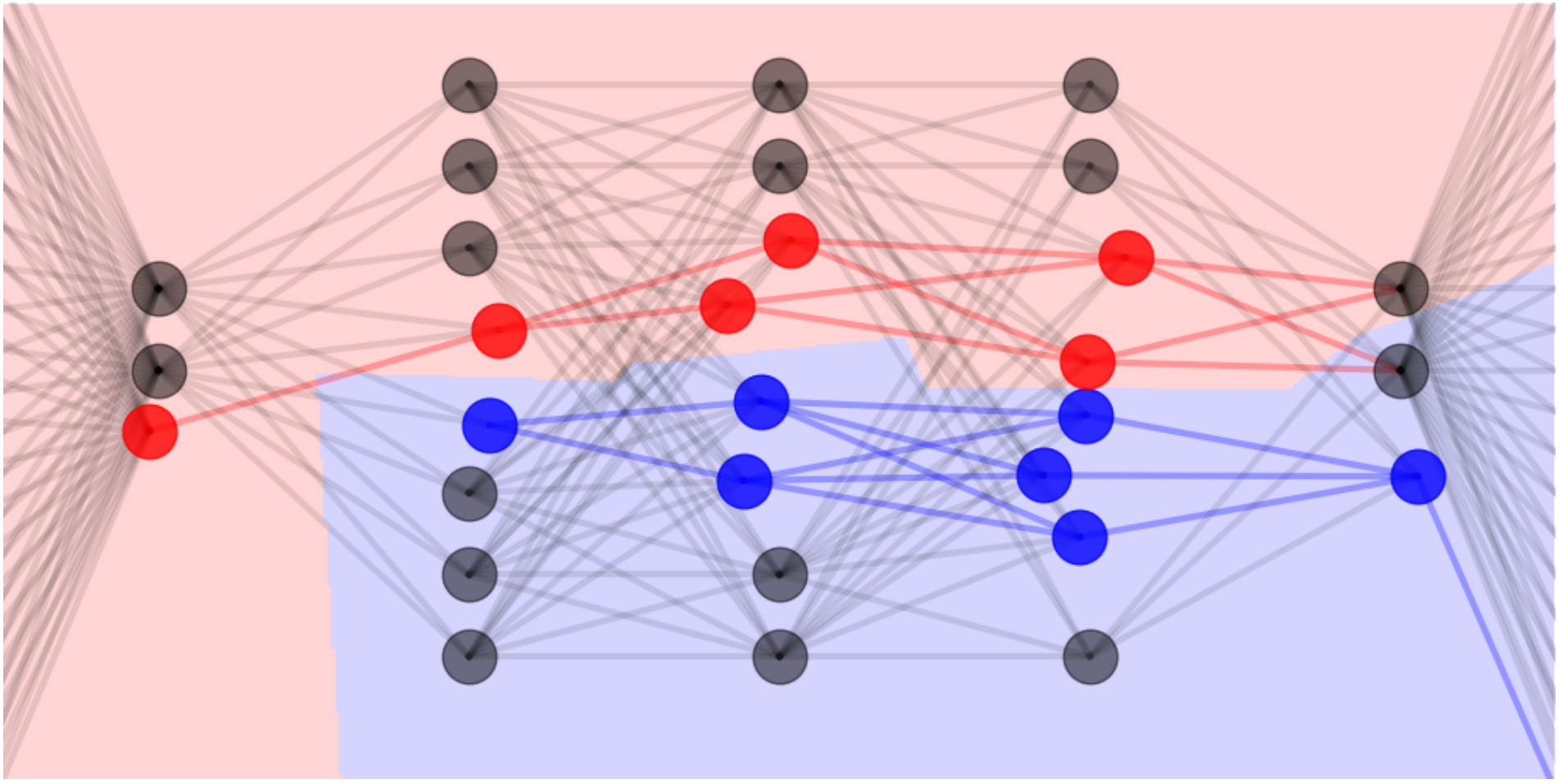
TU-Delft Machine and Deep Learning



Optimization

Lecture 4

Lecturer: Jan van Gemert

Several slides inspired by Andrew Ng

Topic: Optimization

Topics:

- Past gradient statistics
- How to efficiently compute past statistics
- Momentum, RMSprop, Adam.
- Feature and Batch normalization

DL-book chapters: 8.3.1, 8.3.2, 8.5.2, 8.5.3, 8.7.1
UDL-book chapters: 6.3, 6.4.

Training a network

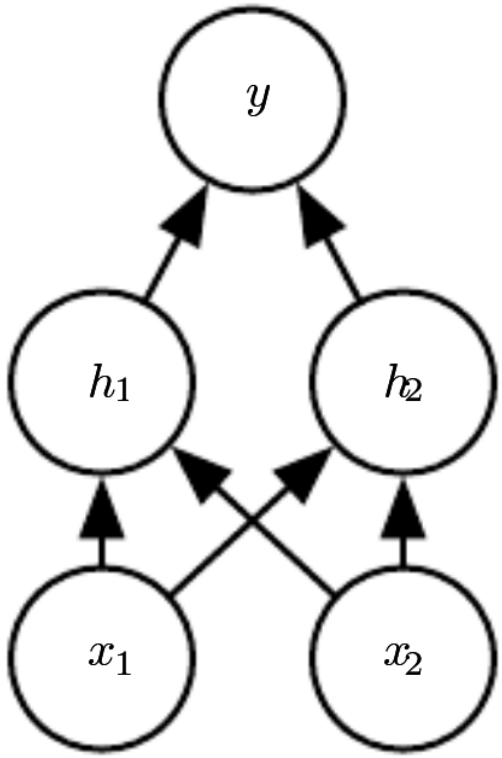
Chapter 4.3

Training set:

Data		Label
0	1	1
1	1	0
0	0	0
1	0	1

While not converged:

1. Present a training sample
2. Compare the results
3. Update the weights



Training a network

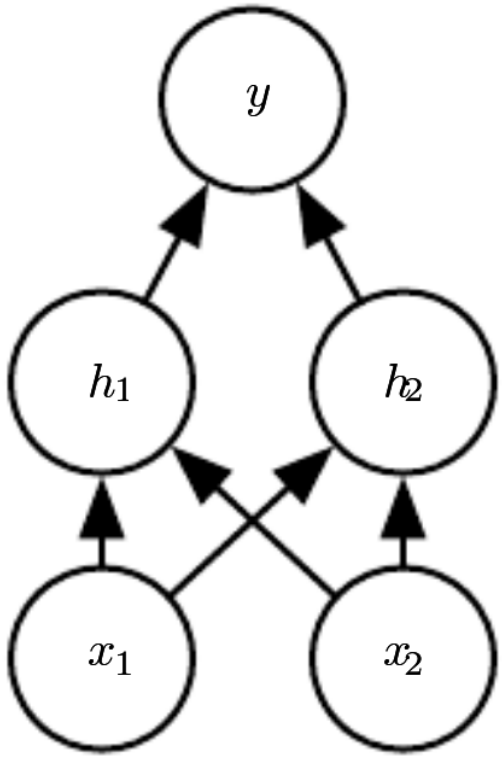
Chapter 4.3

Training set:

Data		Label
0	1	1
1	1	0
0	0	0
1	0	1

While not converged:

1. Present a training sample $\rightarrow$ Get values
2. Compare the results
3. Update the weights



Training a network

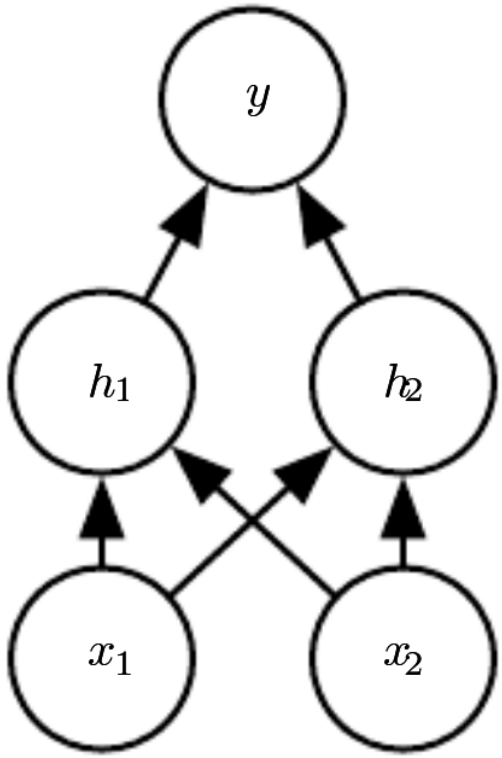
Chapter 4.3

Training set:

Data		Label
0	1	1
1	1	0
0	0	0
1	0	1

While not converged:

1. Present a training sample $\rightarrow$ Get values
2. Compare the results $\rightarrow$ Loss
3. Update the weights



Training a network

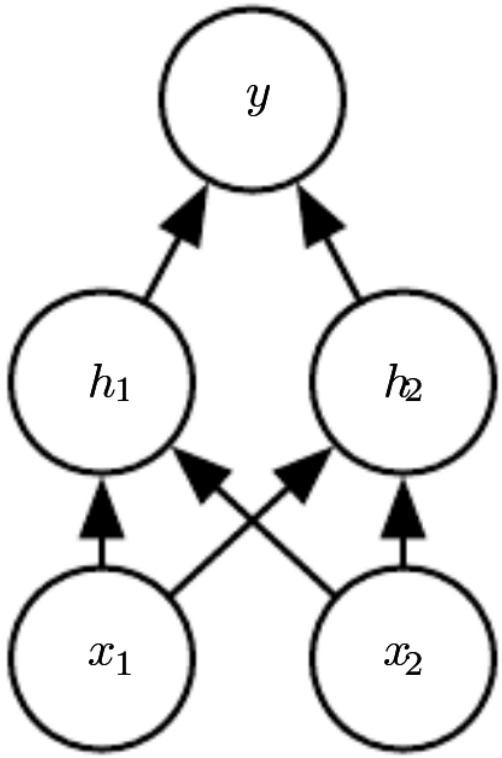
Chapter 4.3

Training set:

Data		Label
0	1	1
1	1	0
0	0	0
1	0	1

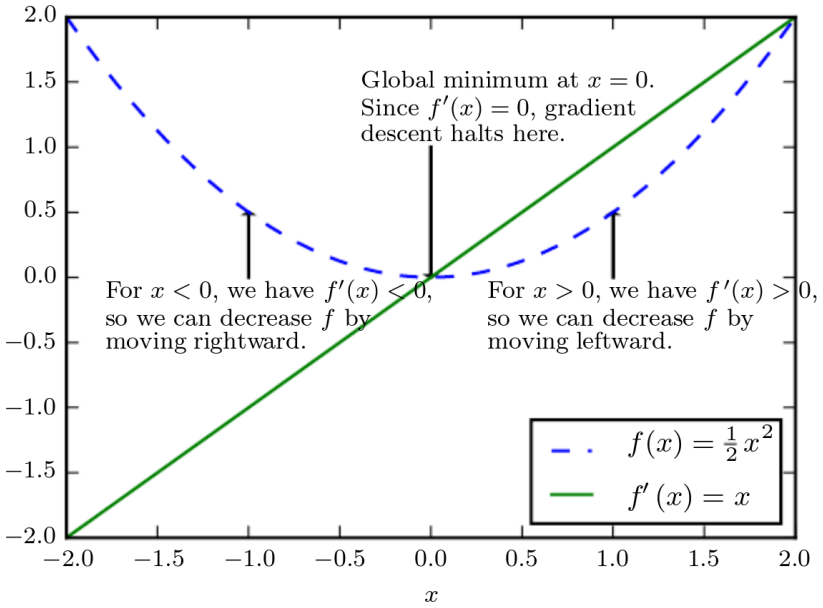
While not converged:

1. Present a training sample $\rightarrow$ Get values
2. Compare the results $\rightarrow$ Loss
3. Update the weights $\rightarrow$ Backprop



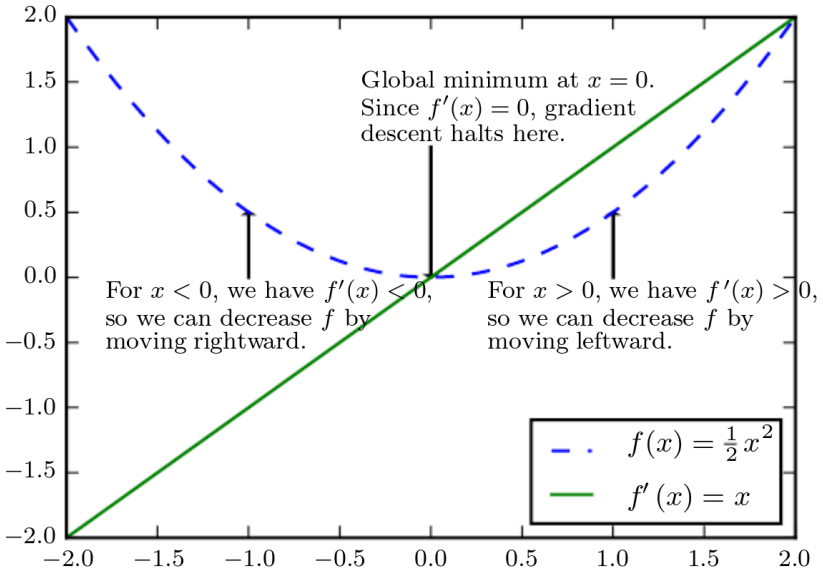
Stochastic Gradient Descent (SGD)

Chapter 5.9 and 8.3.1



Stochastic Gradient Descent (SGD)

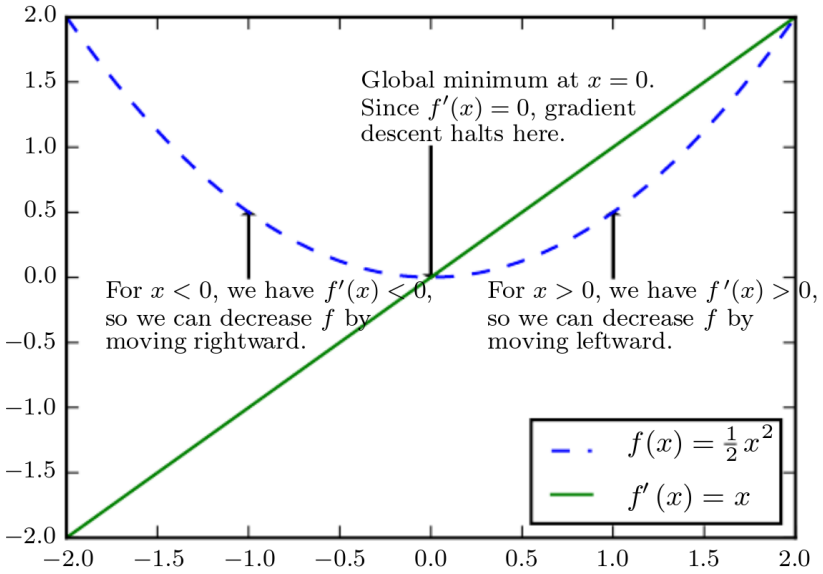
Chapter 5.9 and 8.3.1



- Gradient: Vector of all partial derivatives $\nabla_x f(x)$

Stochastic Gradient Descent (SGD)

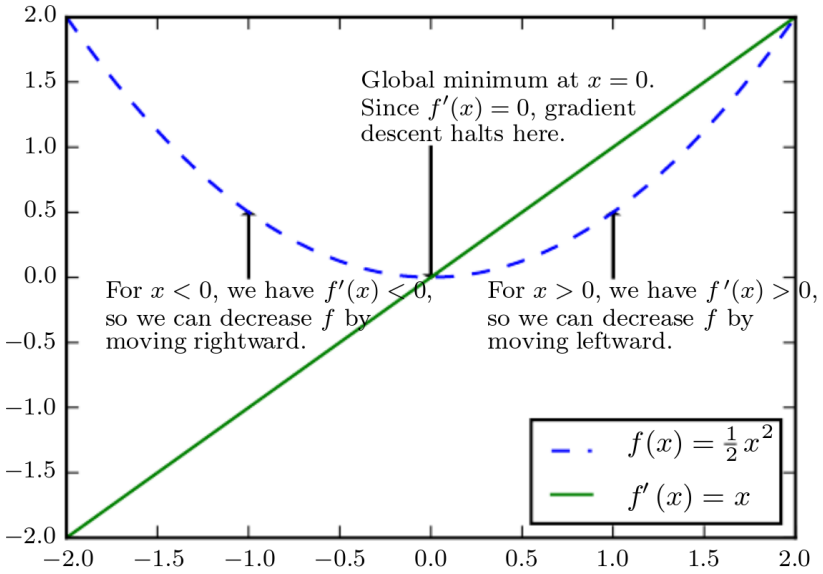
Chapter 5.9 and 8.3.1



- Gradient: Vector of all partial derivatives $\nabla_x f(x)$
- $\theta^* = \theta - \epsilon \nabla_\theta f(x)$:

Stochastic Gradient Descent (SGD)

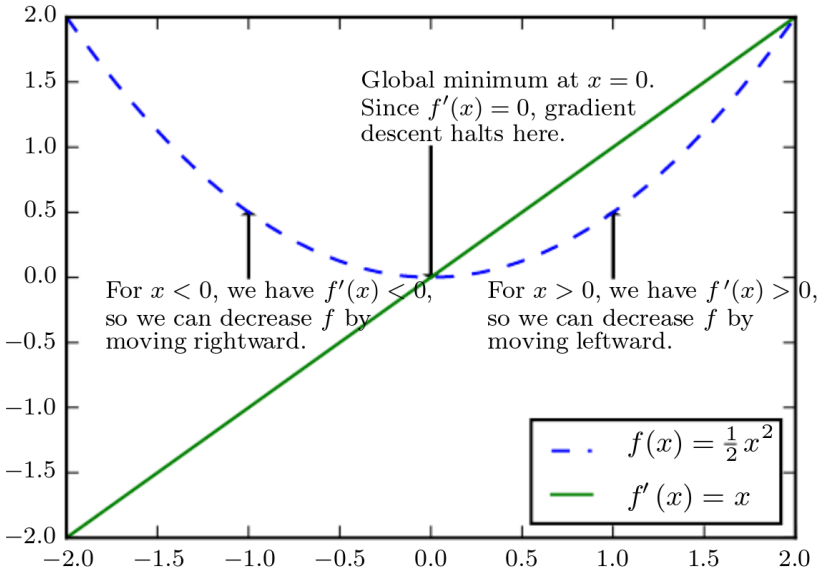
Chapter 5.9 and 8.3.1



- Gradient: Vector of all partial derivatives $\nabla_x f(x)$
- $\theta^* = \theta - \epsilon \nabla_\theta f(x)$: Gradient descent, where ϵ is the learning rate

Stochastic Gradient Descent (SGD)

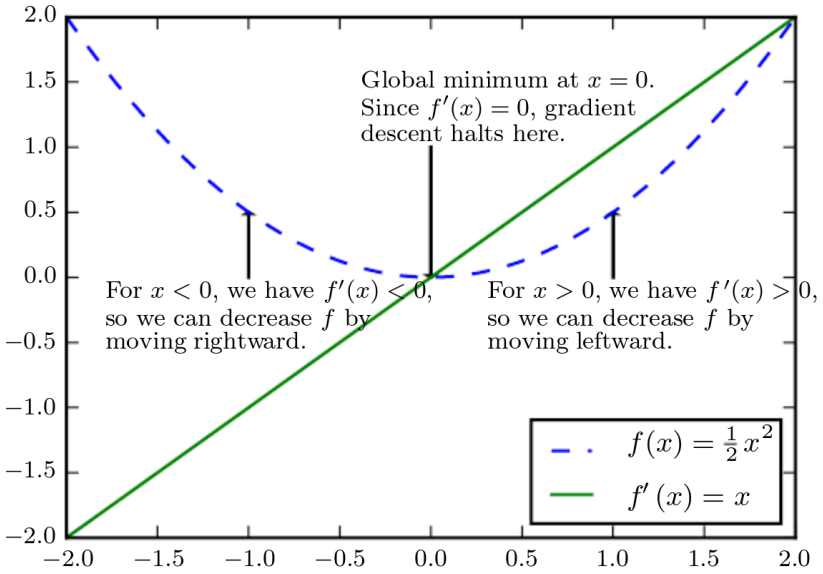
Chapter 5.9 and 8.3.1



- Gradient: Vector of all partial derivatives $\nabla_x f(x)$
- $\theta^* = \theta - \epsilon \nabla_\theta f(x)$: Gradient descent, where ϵ is the learning rate
- Loss all samples: $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} L(x^{(i)}, y^{(i)}, \theta)$

Stochastic Gradient Descent (SGD)

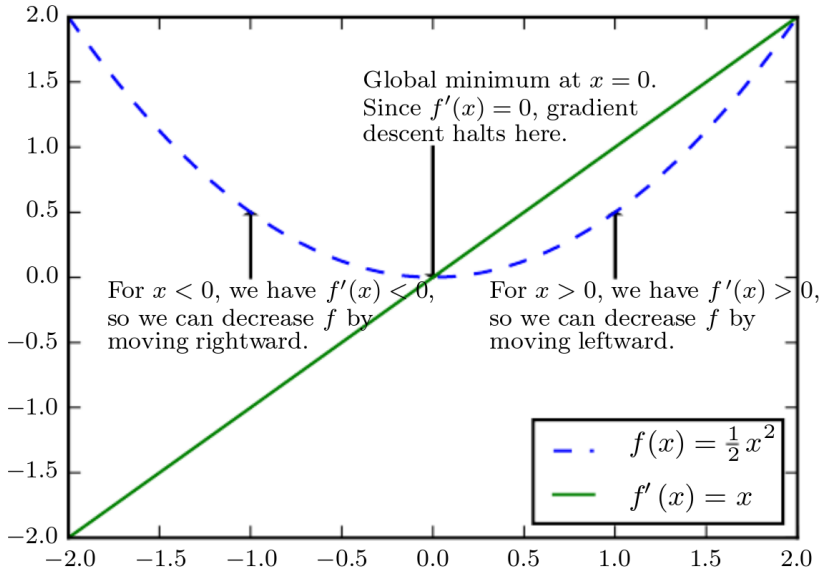
Chapter 5.9 and 8.3.1



- Gradient: Vector of all partial derivatives $\nabla_x f(x)$
- $\theta^* = \theta - \epsilon \nabla_\theta f(x)$: Gradient descent, where ϵ is the learning rate
- Loss all samples: $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} L(x^{(i)}, y^{(i)}, \theta)$
- Gradient all samples: $\nabla_\theta J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta L(x^{(i)}, y^{(i)}, \theta)$

Stochastic Gradient Descent (SGD)

Chapter 5.9 and 8.3.1

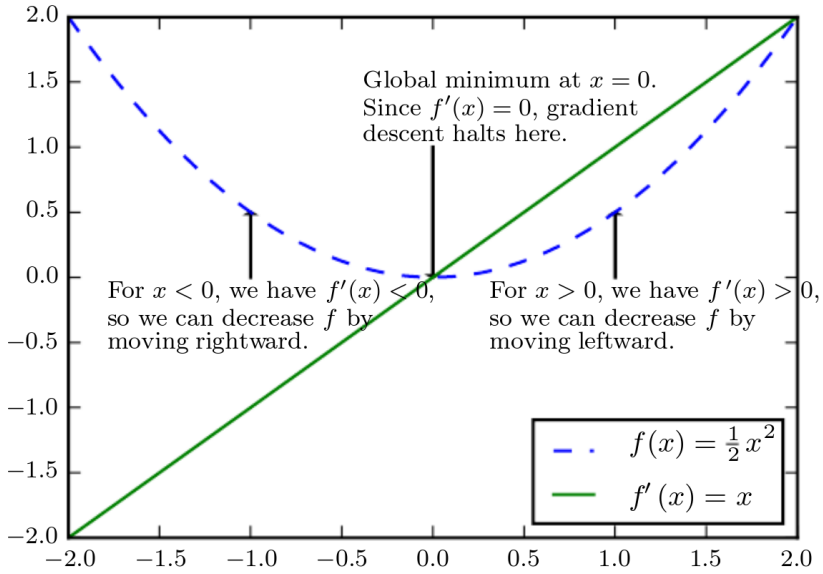


- Gradient: Vector of all partial derivatives $\nabla_x f(x)$
- $\theta^* = \theta - \epsilon \nabla_\theta f(x)$: Gradient descent, where ϵ is the learning rate
- Loss all samples: $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} L(x^{(i)}, y^{(i)}, \theta)$
- Gradient all samples: $\nabla_\theta J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta L(x^{(i)}, y^{(i)}, \theta)$

SGD: mini-batch (k) exact gradient approximating the all-samples (m) gradient:

Stochastic Gradient Descent (SGD)

Chapter 5.9 and 8.3.1



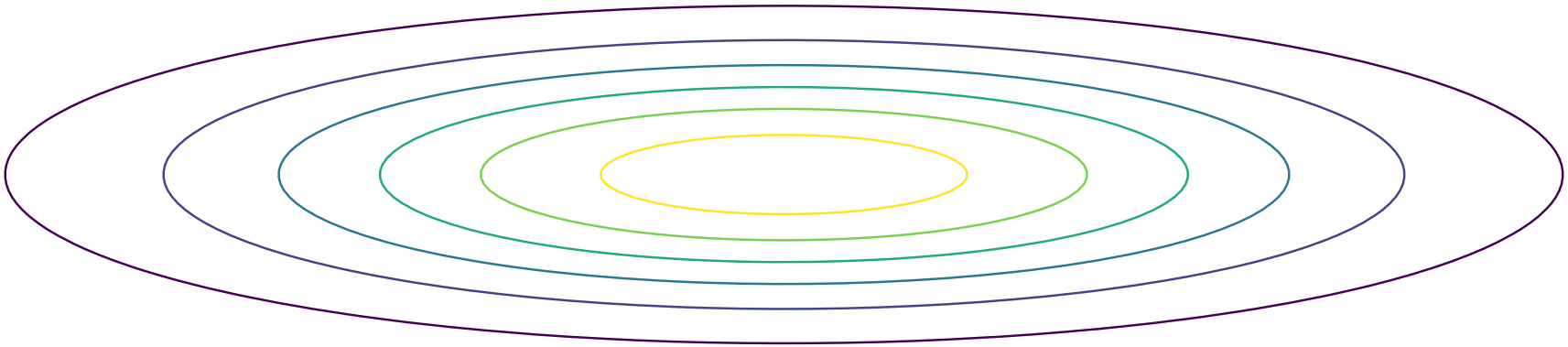
- Gradient: Vector of all partial derivatives $\nabla_x f(x)$
- $\theta^* = \theta - \epsilon \nabla_\theta f(x)$: Gradient descent, where ϵ is the learning rate
- Loss all samples: $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} L(x^{(i)}, y^{(i)}, \theta)$
- Gradient all samples: $\nabla_\theta J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta L(x^{(i)}, y^{(i)}, \theta)$

SGD: mini-batch (k) exact gradient approximating the all-samples (m) gradient:

$$\theta^* = \theta - \epsilon \frac{1}{k} \sum_{i=1}^{k} \nabla_\theta L(x^{(i)}, y^{(i)}, \theta)$$

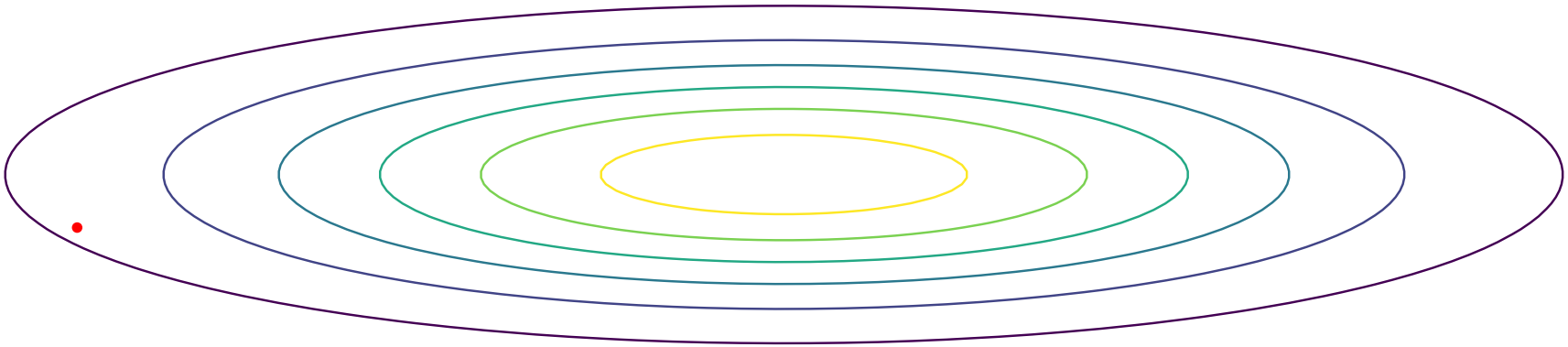
Tuning the learning rate; take 1.

The learning rate is arguably the most important parameter to tune



Tuning the learning rate; take 1.

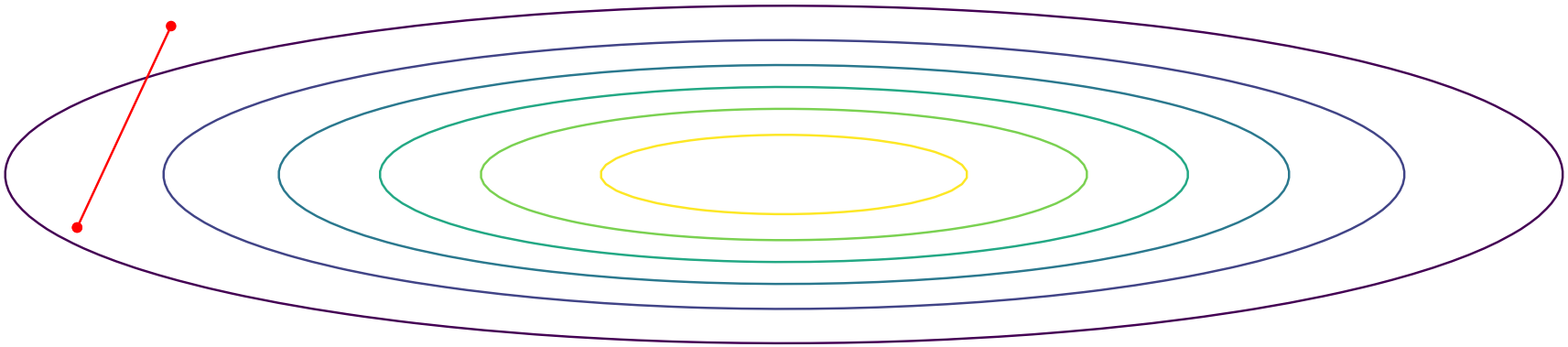
The learning rate is arguably the most important parameter to tune



Tuning the learning rate; take 1.

The learning rate is arguably the most important parameter to tune

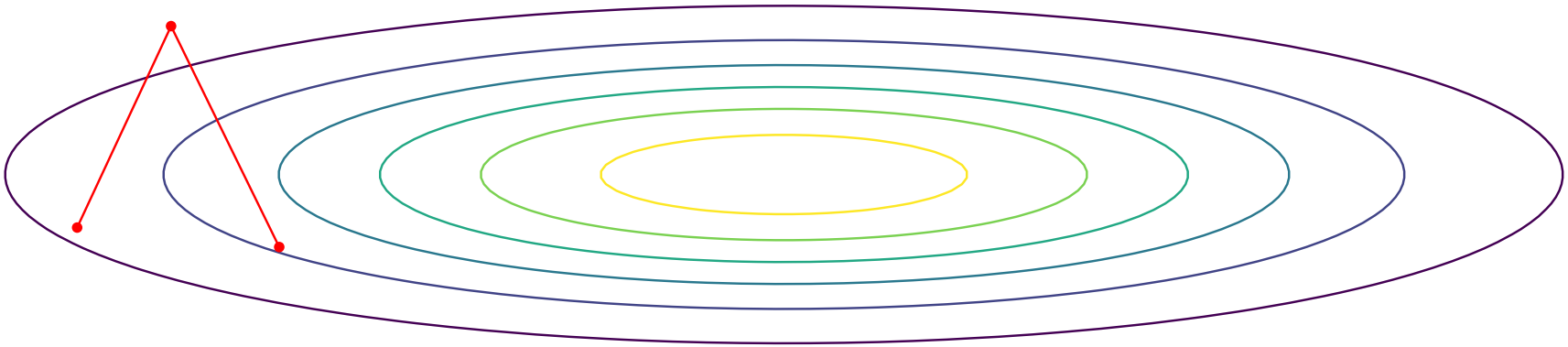
Q: Wait.. What is the ellipse? What are these red dots? Red line?



Tuning the learning rate; take 1.

The learning rate is arguably the most important parameter to tune

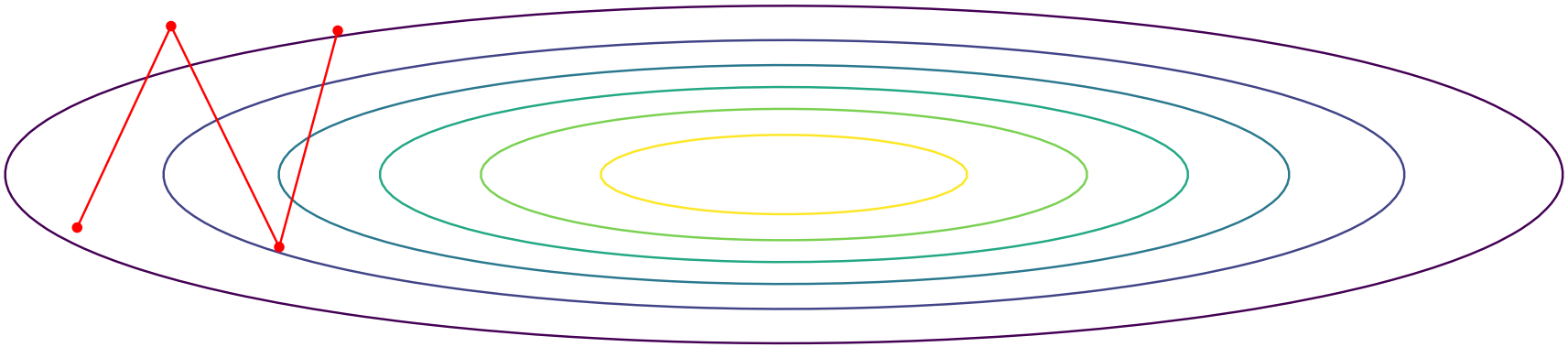
Q: Wait.. What is the ellipse? What are these red dots? Red line?
A: Ellipse is contour plot of the loss for the full 2D space (center is low = good).



Tuning the learning rate; take 1.

The learning rate is arguably the most important parameter to tune

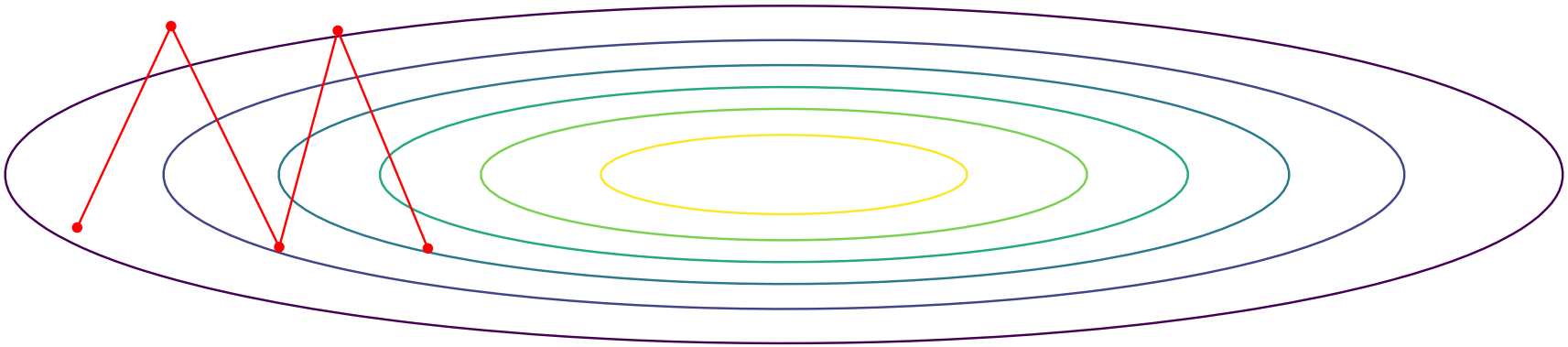
Q: Wait.. What is the ellipse? What are these red dots? Red line?
A: Ellipse is contour plot of the loss for the full 2D space (center is low = good).
A: Each red dot is a particular parameter value for the full net.
(In reality not 2D, but thousands of dimensions: one per parameter)



Tuning the learning rate; take 1.

The learning rate is arguably the most important parameter to tune

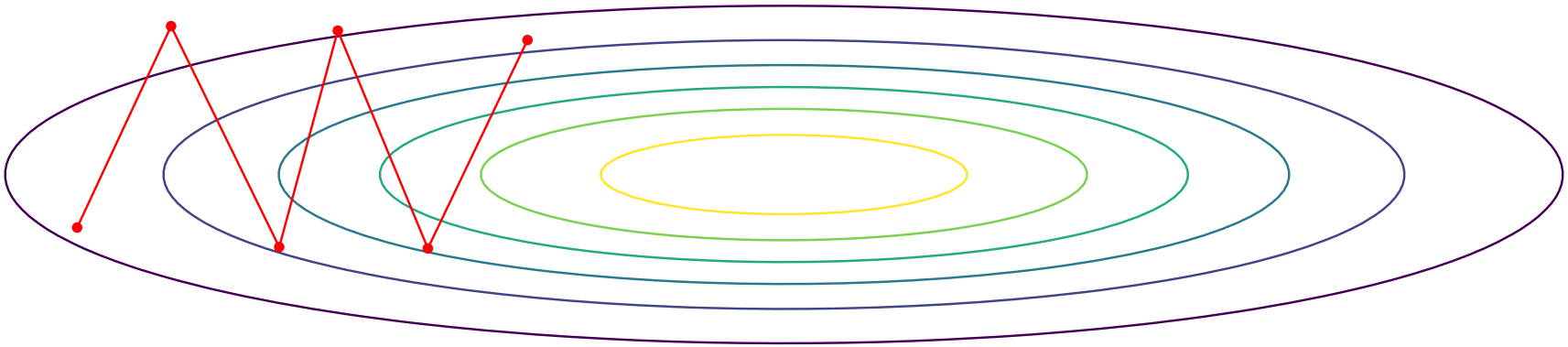
Q: Wait.. What is the ellipse? What are these red dots? Red line?
A: Ellipse is contour plot of the loss for the full 2D space (center is low = good).
A: Each red dot is a particular parameter value for the full net.
(In reality not 2D, but thousands of dimensions: one per parameter)
A: The red line is the trajectory of weight updated through SGD.



Tuning the learning rate; take 1.

The learning rate is arguably the most important parameter to tune

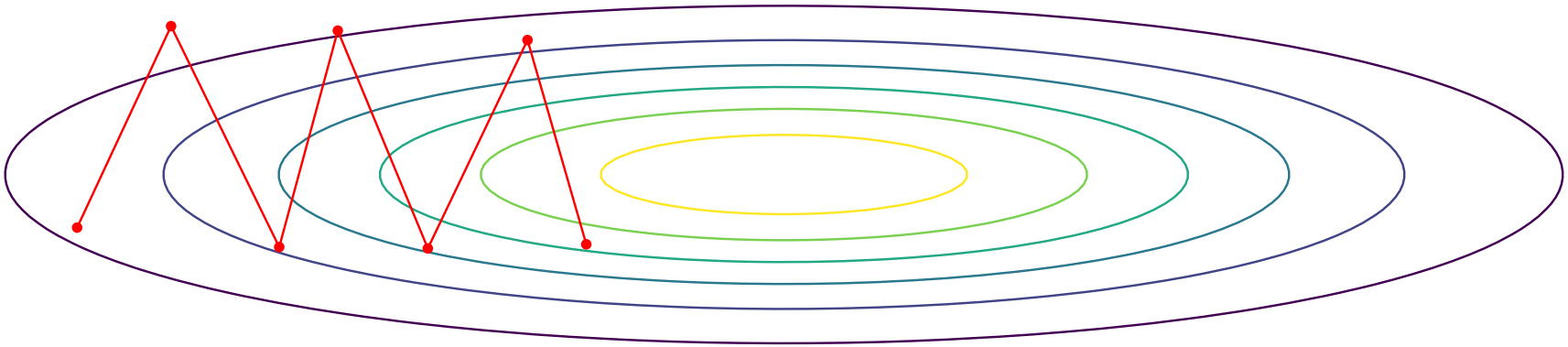
Q: Wait.. What is the ellipse? What are these red dots? Red line?
A: Ellipse is contour plot of the loss for the full 2D space (center is low = good).
A: Each red dot is a particular parameter value for the full net.
(In reality not 2D, but thousands of dimensions: one per parameter)
A: The red line is the trajectory of weight updated through SGD.



Tuning the learning rate; take 1.

The learning rate is arguably the most important parameter to tune

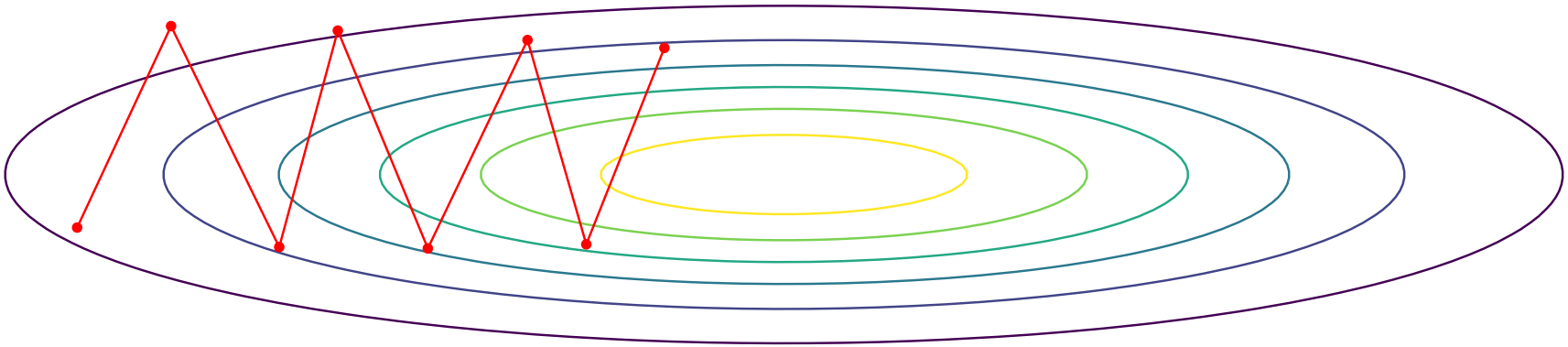
Q: Wait.. What is the ellipse? What are these red dots? Red line?
A: Ellipse is contour plot of the loss for the full 2D space (center is low = good).
A: Each red dot is a particular parameter value for the full net.
(In reality not 2D, but thousands of dimensions: one per parameter)
A: The red line is the trajectory of weight updated through SGD.



Tuning the learning rate; take 1.

The learning rate is arguably the most important parameter to tune

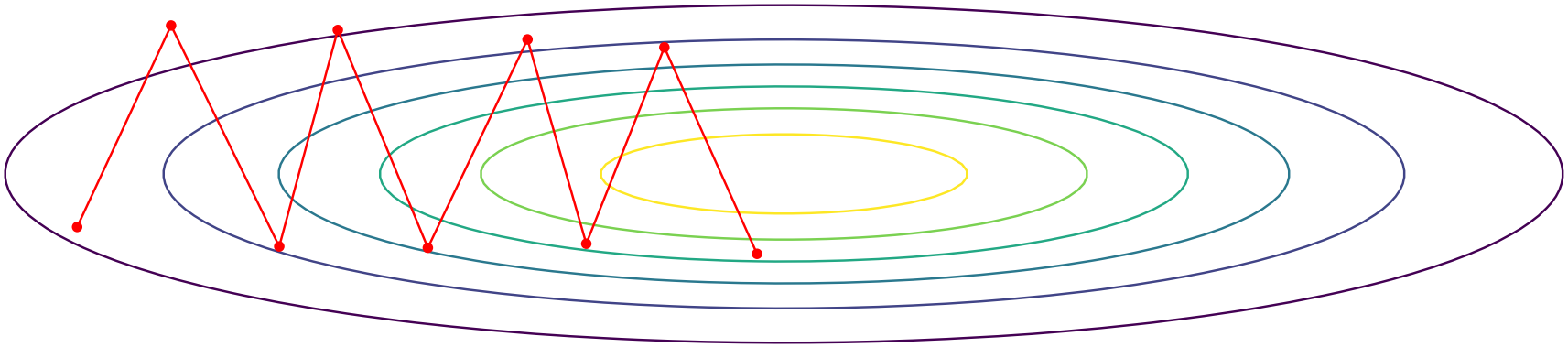
Q: Wait.. What is the ellipse? What are these red dots? Red line?
A: Ellipse is contour plot of the loss for the full 2D space (center is low = good).
A: Each red dot is a particular parameter value for the full net.
(In reality not 2D, but thousands of dimensions: one per parameter)
A: The red line is the trajectory of weight updated through SGD.



Tuning the learning rate; take 1.

The learning rate is arguably the most important parameter to tune

Q: Wait.. What is the ellipse? What are these red dots? Red line?
A: Ellipse is contour plot of the loss for the full 2D space (center is low = good).
A: Each red dot is a particular parameter value for the full net.
(In reality not 2D, but thousands of dimensions: one per parameter)
A: The red line is the trajectory of weight updated through SGD.

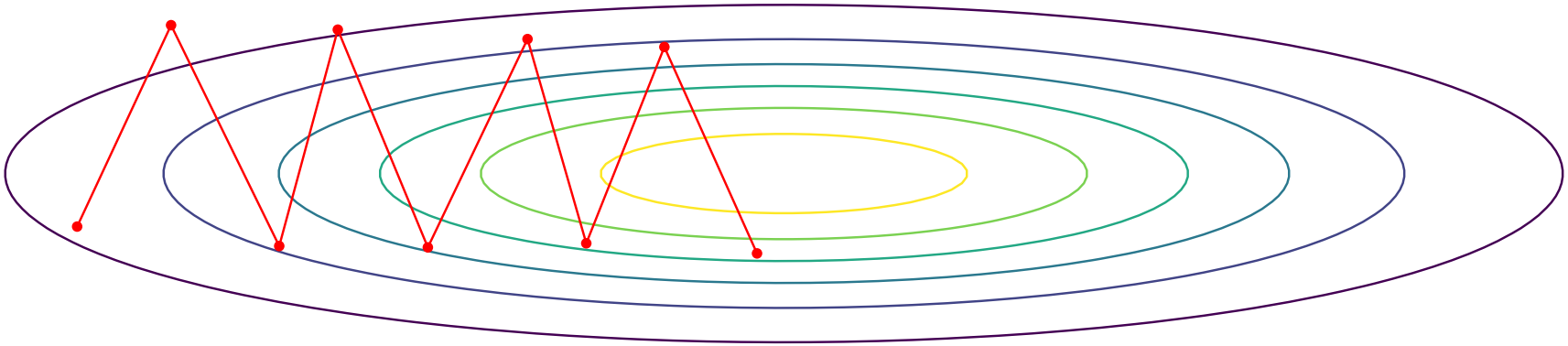


Q: What do you notice?

Tuning the learning rate; take 1.

The learning rate is arguably the most important parameter to tune

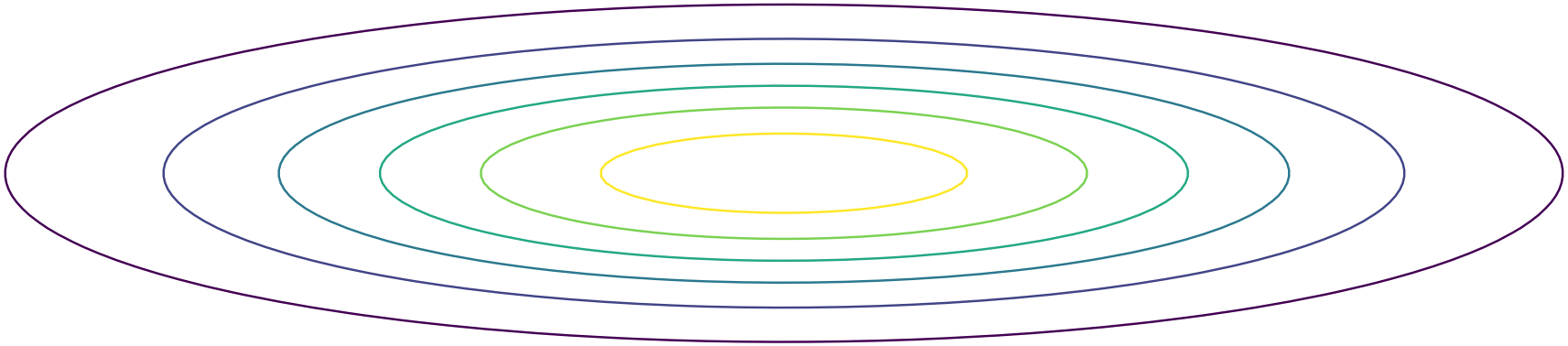
Q: Wait.. What is the ellipse? What are these red dots? Red line?
A: Ellipse is contour plot of the loss for the full 2D space (center is low = good).
A: Each red dot is a particular parameter value for the full net.
(In reality not 2D, but thousands of dimensions: one per parameter)
A: The red line is the trajectory of weight updated through SGD.



Q: What do you notice?
A: If LR too high it will never find a good instantiation

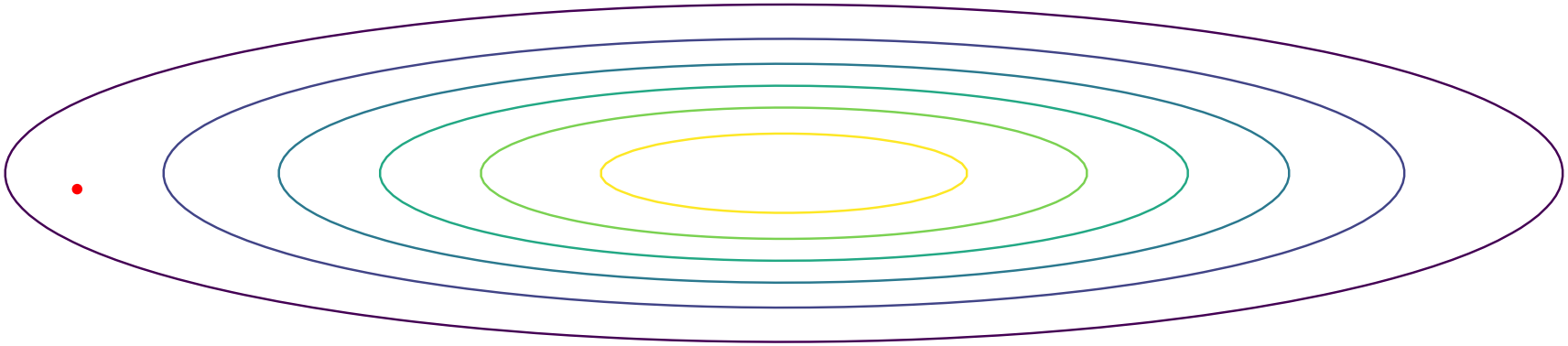
Tuning the learning rate; take 2.

The learning rate is arguably the most important parameter to tune



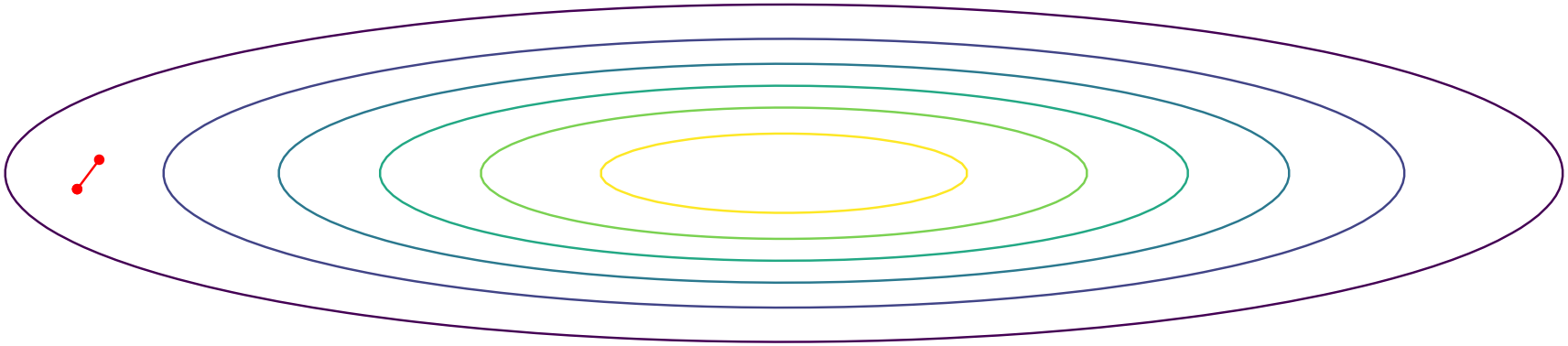
Tuning the learning rate; take 2.

The learning rate is arguably the most important parameter to tune



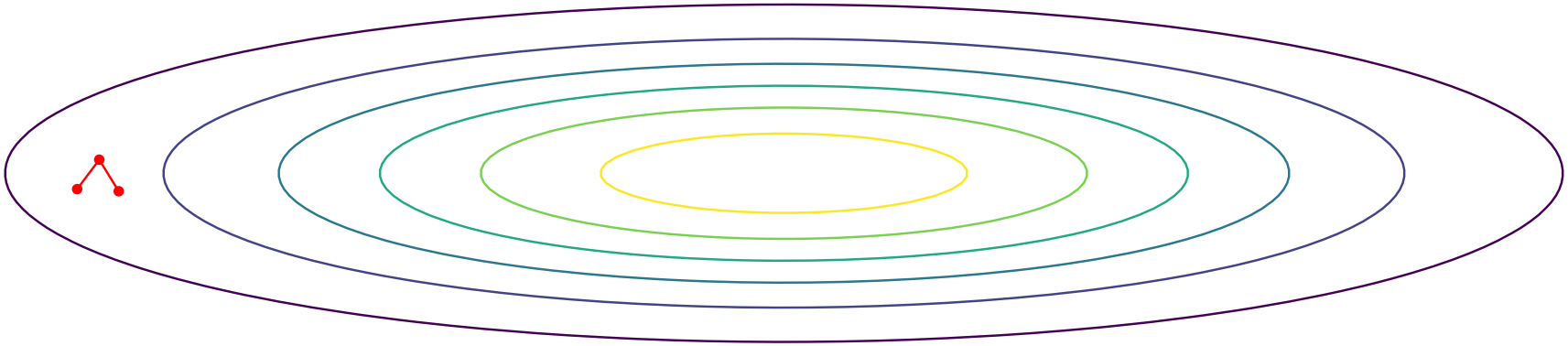
Tuning the learning rate; take 2.

The learning rate is arguably the most important parameter to tune



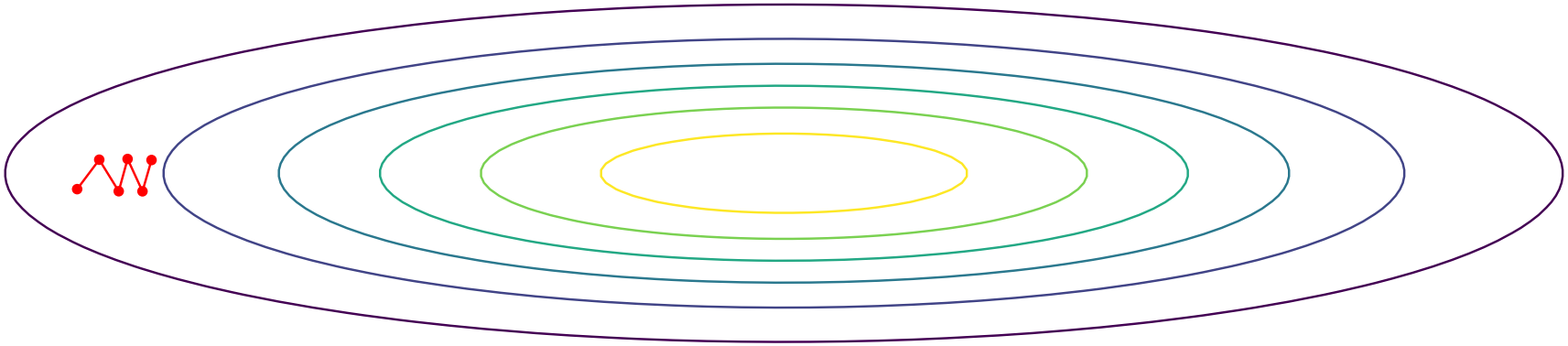
Tuning the learning rate; take 2.

The learning rate is arguably the most important parameter to tune



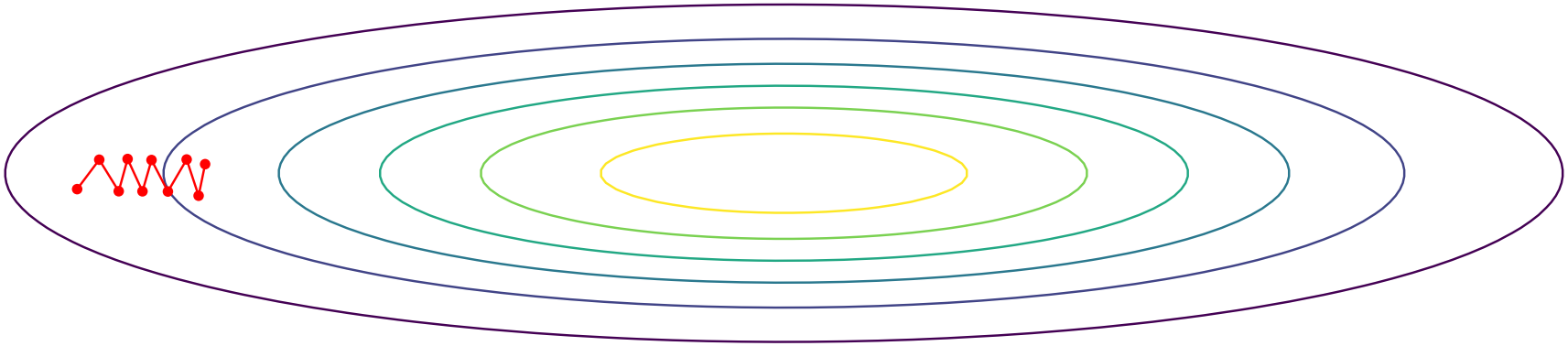
Tuning the learning rate; take 2.

The learning rate is arguably the most important parameter to tune



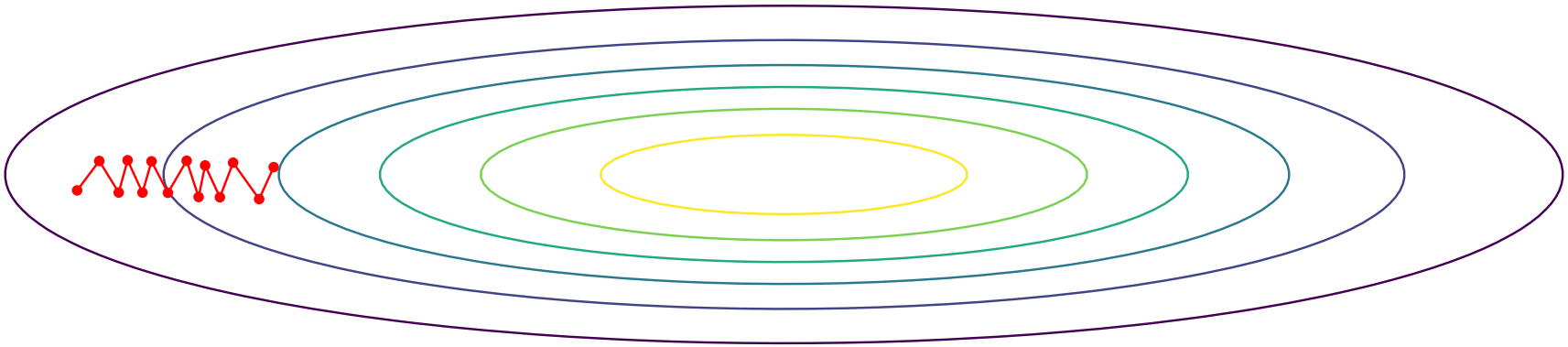
Tuning the learning rate; take 2.

The learning rate is arguably the most important parameter to tune



Tuning the learning rate; take 2.

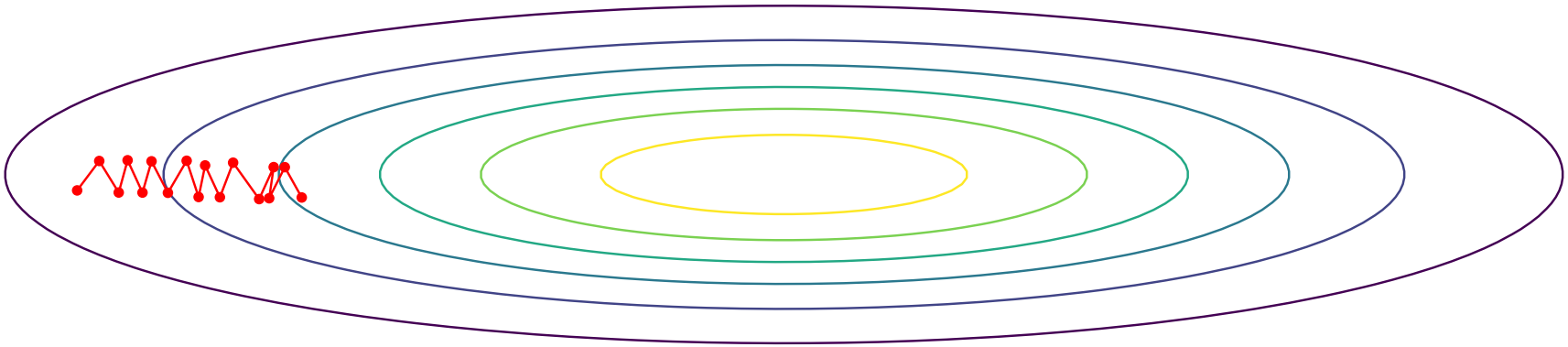
The learning rate is arguably the most important parameter to tune



Q: What do you notice?

Tuning the learning rate; take 2.

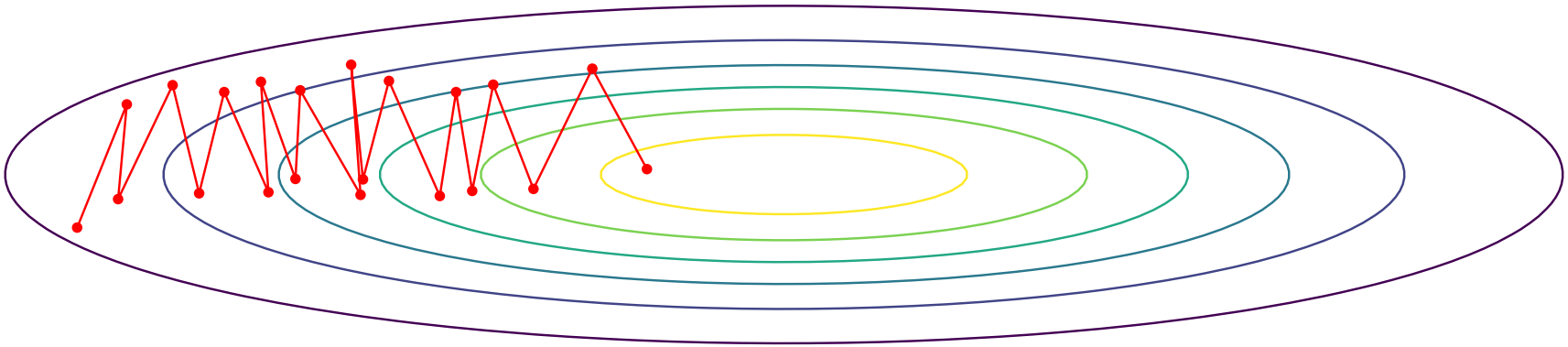
The learning rate is arguably the most important parameter to tune



A: If LR too low it will take very long to find a good instantiation

Tuning the learning rate; take 3.

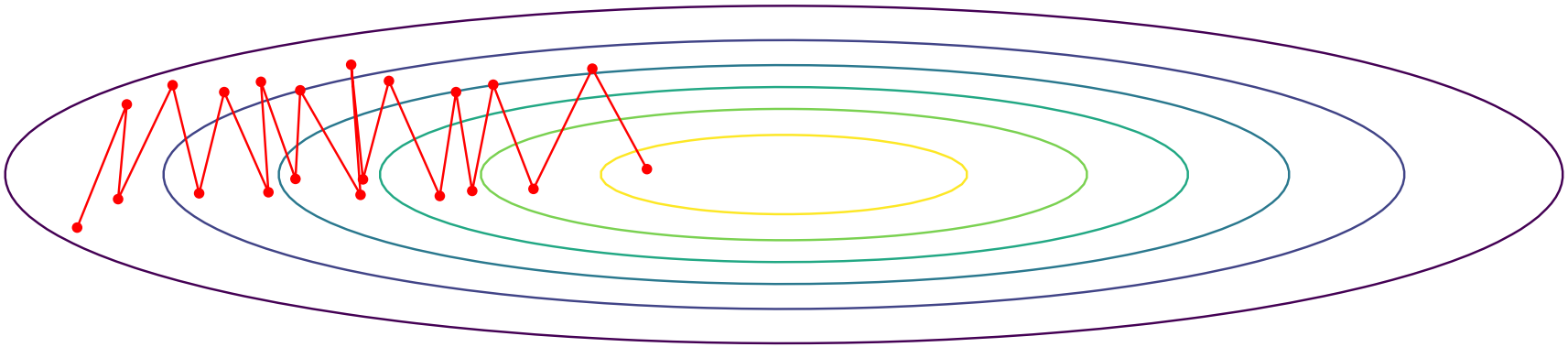
The learning rate is arguably the most important parameter to tune



Q: What do you notice?

Tuning the learning rate; take 3.

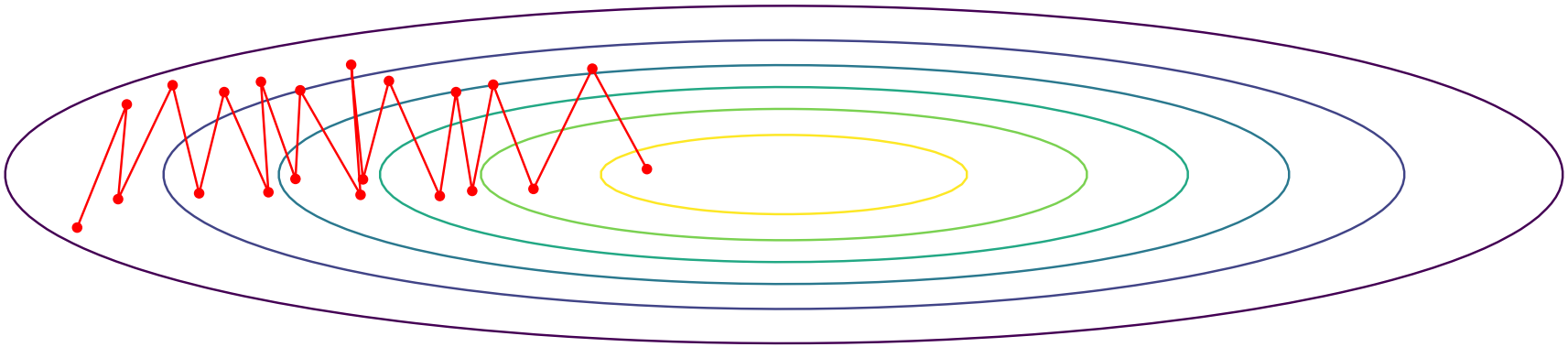
The learning rate is arguably the most important parameter to tune



Q: What do you notice? A: SGD is noisy; it's also noisy close to a 'good place'.

Tuning the learning rate; take 3.

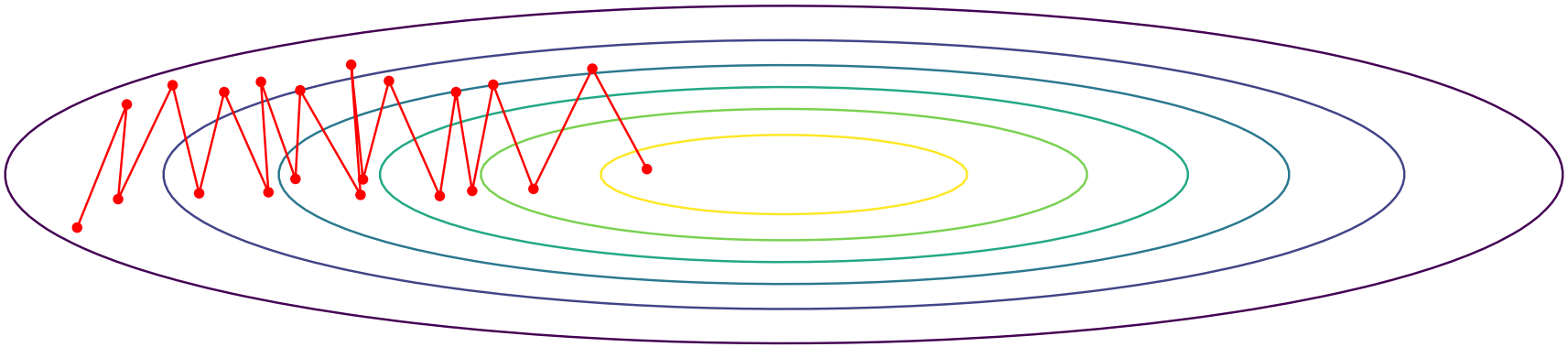
The learning rate is arguably the most important parameter to tune



Q: What do you notice? A: SGD is noisy; it's also noisy close to a 'good place'.
Q: How to take smaller steps?

Tuning the learning rate; take 3.

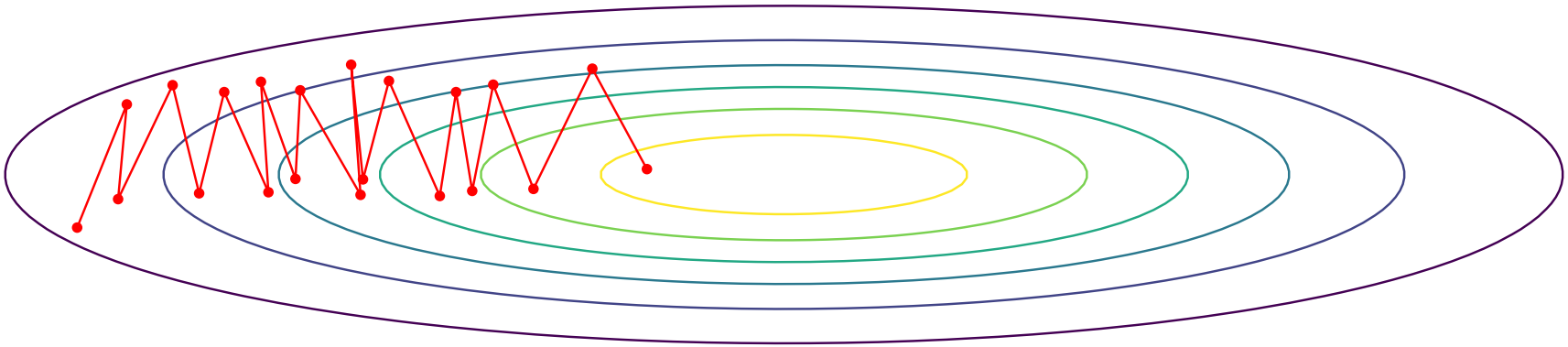
The learning rate is arguably the most important parameter to tune



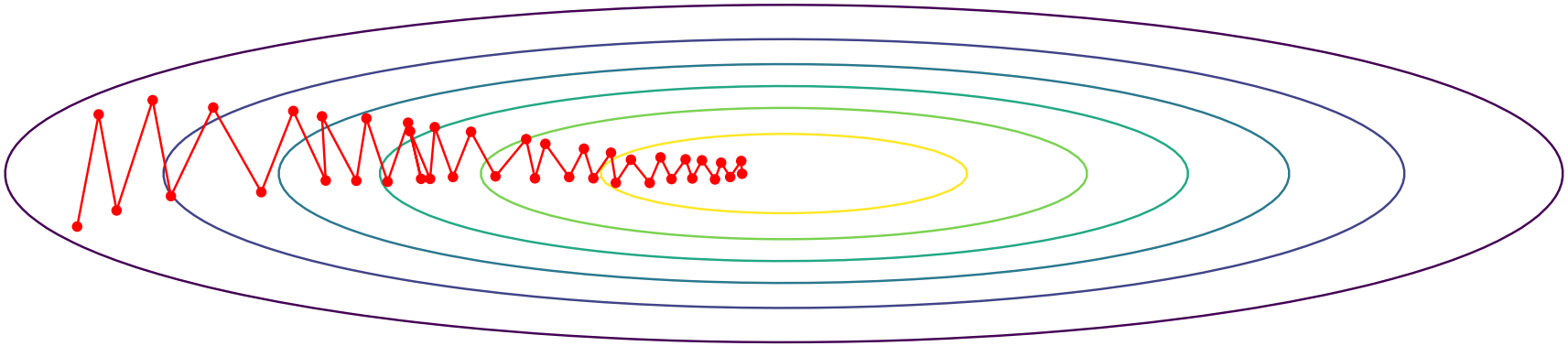
Q: What do you notice? A: SGD is noisy; it's also noisy close to a 'good place'.
Q: How to take smaller steps? A: Reduce the LR over time (decay).

Tuning the learning rate; take 3.

The learning rate is arguably the most important parameter to tune



Q: What do you notice? A: SGD is noisy; it's also noisy close to a 'good place'.
Q: How to take smaller steps? A: Reduce the LR over time (decay).



Questions?

Learning rate: More art than science

DL-Book, Ch. 8.3.1: "Most guidance should be regarded with some skepticism".

Learning rate: More art than science

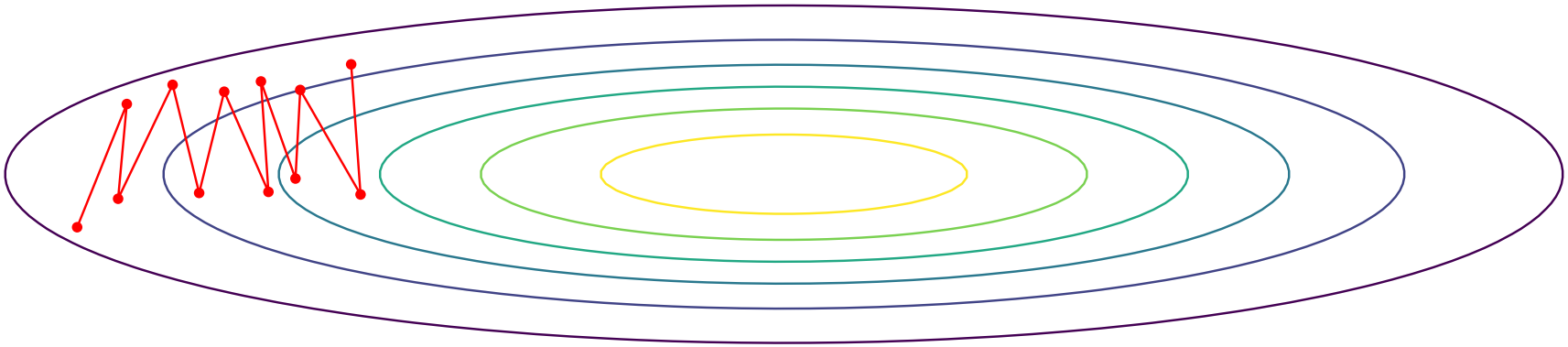
DL-Book, Ch. 8.3.1: "Most guidance should be regarded with some skepticism".

Some algorithms seem to, often, improve over SGD; we discuss 3 such algorithms

Learning rate: More art than science

DL-Book, Ch. 8.3.1: "Most guidance should be regarded with some skepticism".

Some algorithms seem to, often, improve over SGD; we discuss 3 such algorithms

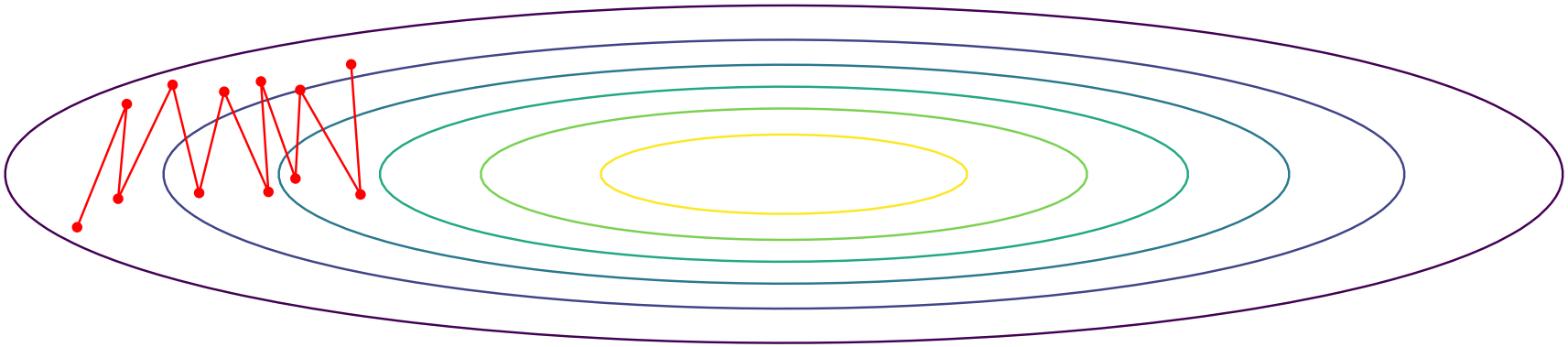


Q: Would you use the same learning rate (LR) for x-axis and y-axis parameters?

Learning rate: More art than science

DL-Book, Ch. 8.3.1: "Most guidance should be regarded with some skepticism".

Some algorithms seem to, often, improve over SGD; we discuss 3 such algorithms

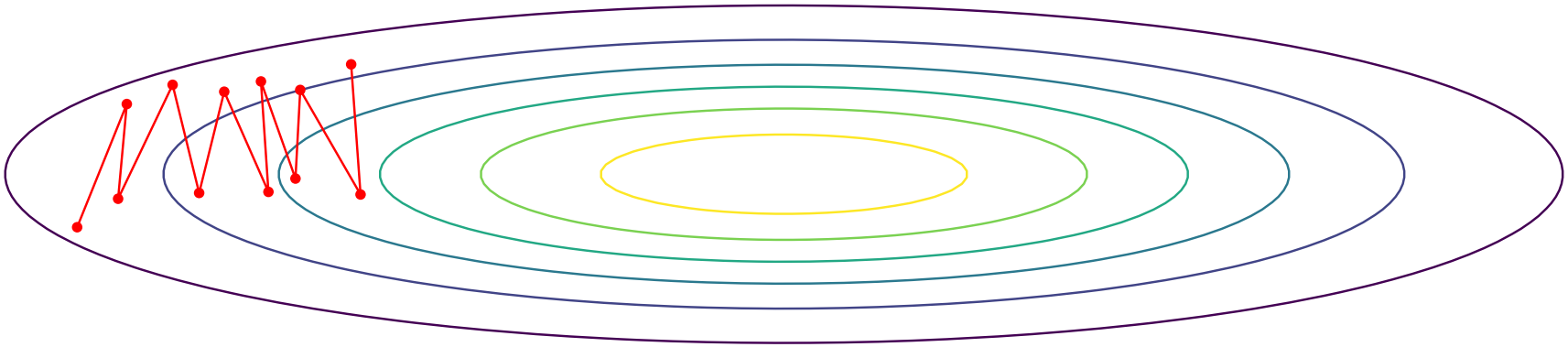


Q: Would you use the same learning rate (LR) for x-axis and y-axis parameters?
A: Use different learning rates per parameter: Smaller for y-axis parameter.

Learning rate: More art than science

DL-Book, Ch. 8.3.1: "Most guidance should be regarded with some skepticism".

Some algorithms seem to, often, improve over SGD; we discuss 3 such algorithms



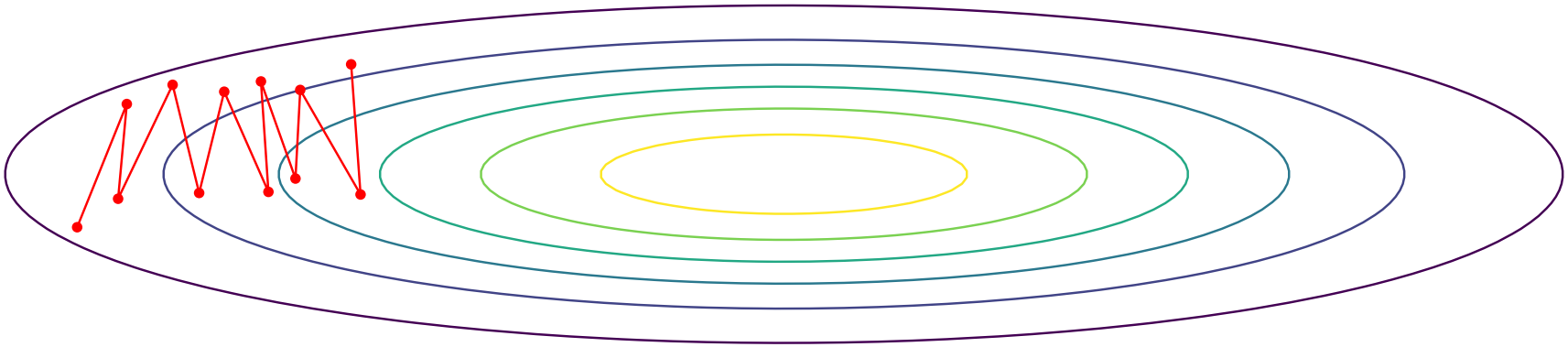
Q: Would you use the same learning rate (LR) for x-axis and y-axis parameters?
A: Use different learning rates per parameter: Smaller for y-axis parameter.

Q: Looking at the history, what about the update direction for x-axis/y-axis?

Learning rate: More art than science

DL-Book, Ch. 8.3.1: "Most guidance should be regarded with some skepticism".

Some algorithms seem to, often, improve over SGD; we discuss 3 such algorithms



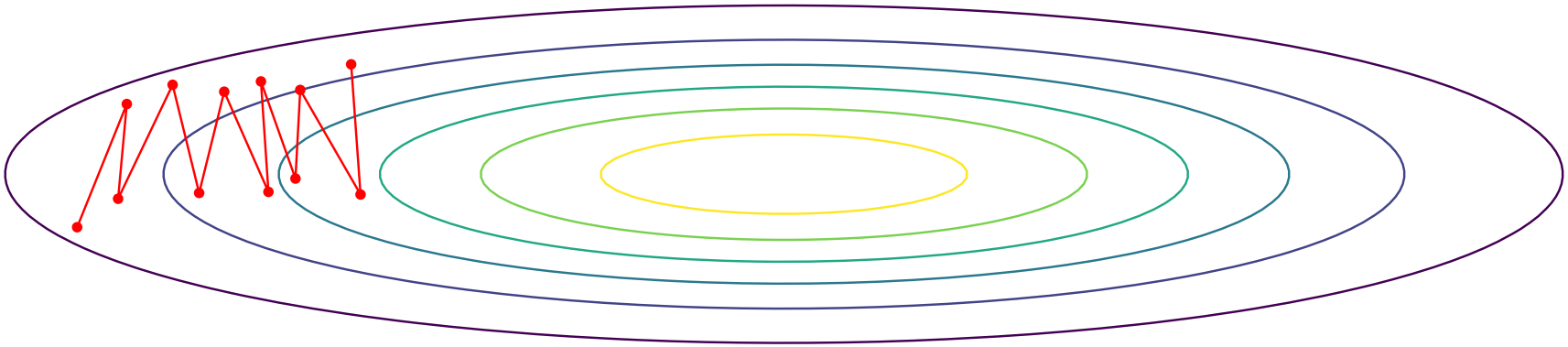
Q: Would you use the same learning rate (LR) for x-axis and y-axis parameters?
A: Use different learning rates per parameter: Smaller for y-axis parameter.

Q: Looking at the history, what about the update direction for x-axis/y-axis?
Momentum algorithm: larger LR if past partial derivative directions are similar.

Learning rate: More art than science

DL-Book, Ch. 8.3.1: "Most guidance should be regarded with some skepticism".

Some algorithms seem to, often, improve over SGD; we discuss 3 such algorithms



Q: Would you use the same learning rate (LR) for x-axis and y-axis parameters?
A: Use different learning rates per parameter: Smaller for y-axis parameter.

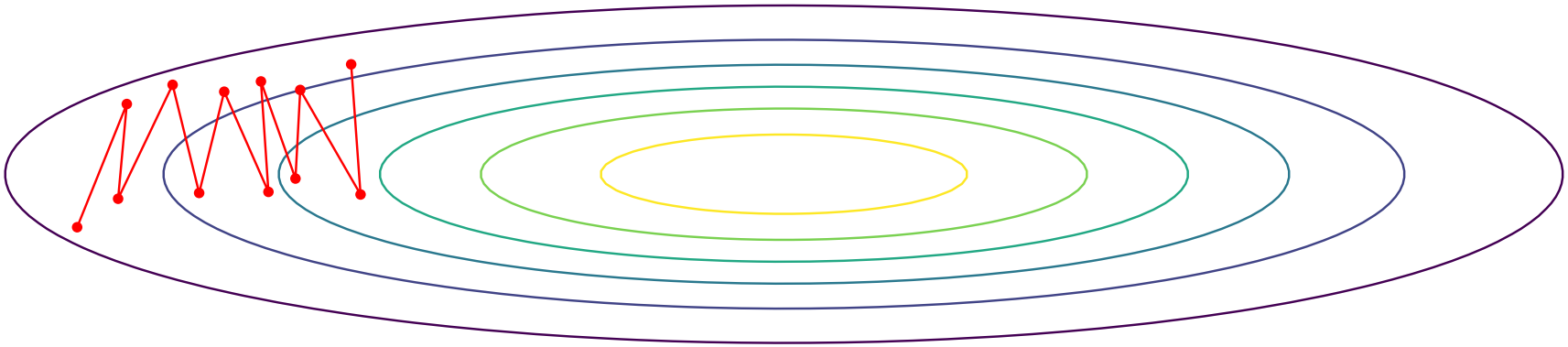
Q: Looking at the history, what about the update direction for x-axis/y-axis?
Momentum algorithm: larger LR if past partial derivative directions are similar.

Q: Looking at the history, what about the update step-size for x-axis/y-axis?

Learning rate: More art than science

DL-Book, Ch. 8.3.1: "Most guidance should be regarded with some skepticism".

Some algorithms seem to, often, improve over SGD; we discuss 3 such algorithms



Q: Would you use the same learning rate (LR) for x-axis and y-axis parameters?
A: Use different learning rates per parameter: Smaller for y-axis parameter.

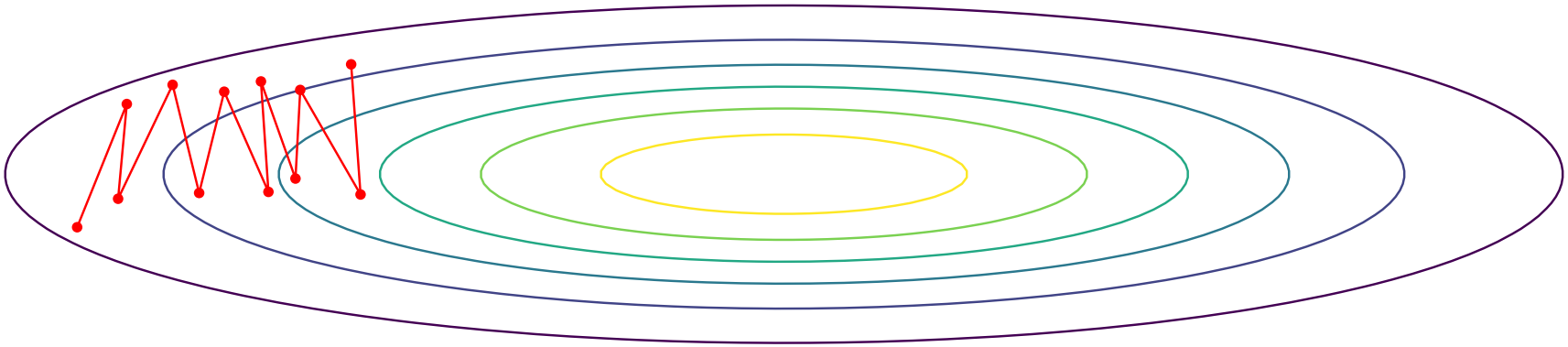
Q: Looking at the history, what about the update direction for x-axis/y-axis?
Momentum algorithm: larger LR if past partial derivative directions are similar.

Q: Looking at the history, what about the update step-size for x-axis/y-axis?
RMSprop algorithm: smaller LR if past partial derivative magnitudes are large.

Learning rate: More art than science

DL-Book, Ch. 8.3.1: "Most guidance should be regarded with some skepticism".

Some algorithms seem to, often, improve over SGD; we discuss 3 such algorithms



Q: Would you use the same learning rate (LR) for x-axis and y-axis parameters?
A: Use different learning rates per parameter: Smaller for y-axis parameter.

Q: Looking at the history, what about the update direction for x-axis/y-axis?
Momentum algorithm: larger LR if past partial derivative directions are similar.

Q: Looking at the history, what about the update step-size for x-axis/y-axis?
RMSprop algorithm: smaller LR if past partial derivative magnitudes are large.

Adam algorithm: combines momentum and RMSprop.

Questions?

Questions?

Useful algorithms:

- Momentum: larger LR if past partial derivative directions are similar.
- RMSprop: smaller LR if past partial derivative magnitudes are large.
- Adam: combines momentum and RMSprop.

Questions?

Useful algorithms:

- Momentum: larger LR if past partial derivative directions are similar.
- RMSprop: smaller LR if past partial derivative magnitudes are large.
- Adam: combines momentum and RMSprop.

All these algorithms make use of past gradient statistics.

Questions?

Useful algorithms:

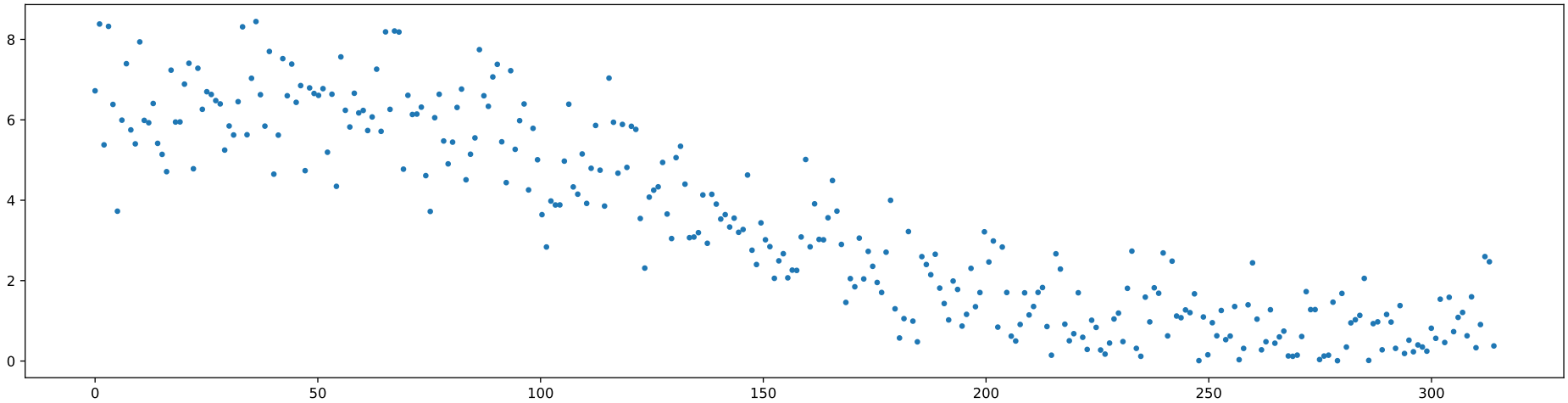
- Momentum: larger LR if past partial derivative directions are similar.
- RMSprop: smaller LR if past partial derivative magnitudes are large.
- Adam: combines momentum and RMSprop.

All these algorithms make use of past gradient statistics.

Next slides: How to efficiently keep track of such statistics.

See a gradients as noisy time series

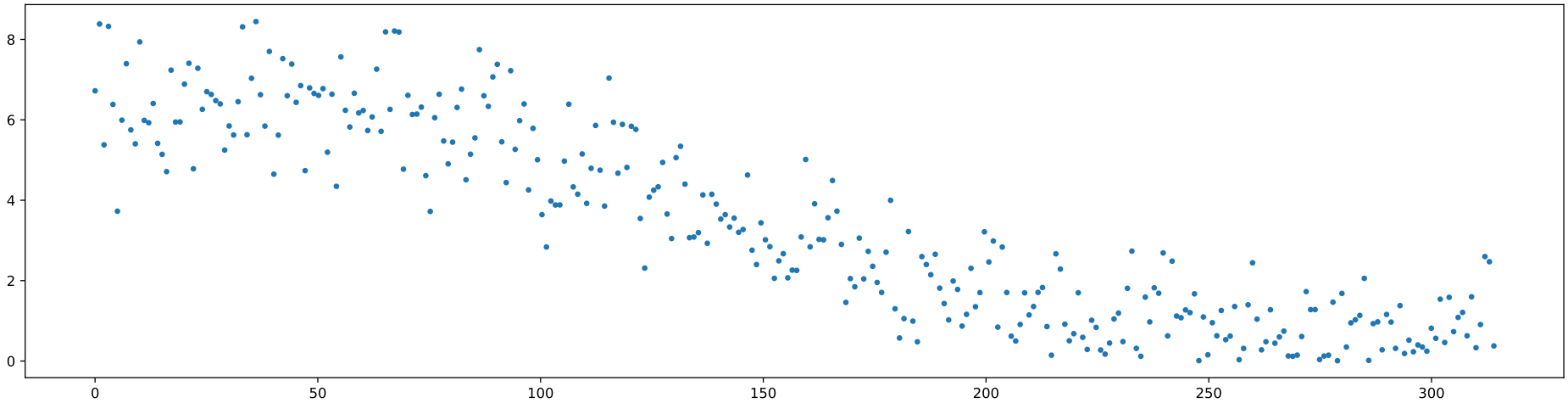
A blue dot is a 1d gradient (y-axis: derivative value) over iterations (x-axis: time series).



How to average-out noisy derivatives as values are coming in over time.

See a gradients as noisy time series

A blue dot is a 1d gradient (y-axis: derivative value) over iterations (x-axis: time series).

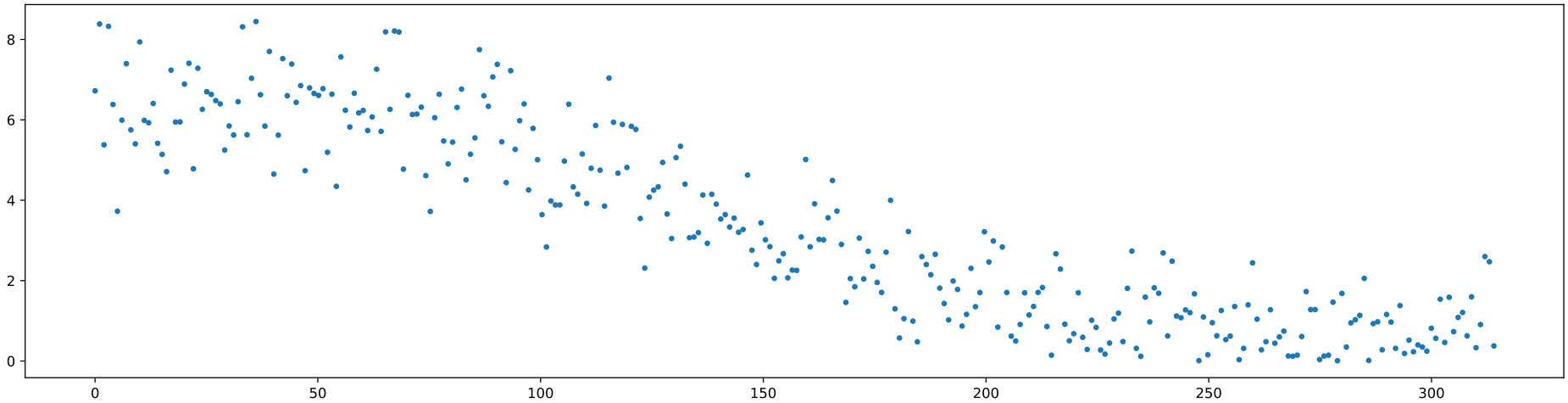


How to average-out noisy derivatives as values are coming in over time.

- Q: Weight each past derivative equally? Eg 10 steps ago vs 100 steps ago?

See a gradients as noisy time series

A blue dot is a 1d gradient (y-axis: derivative value) over iterations (x-axis: time series).

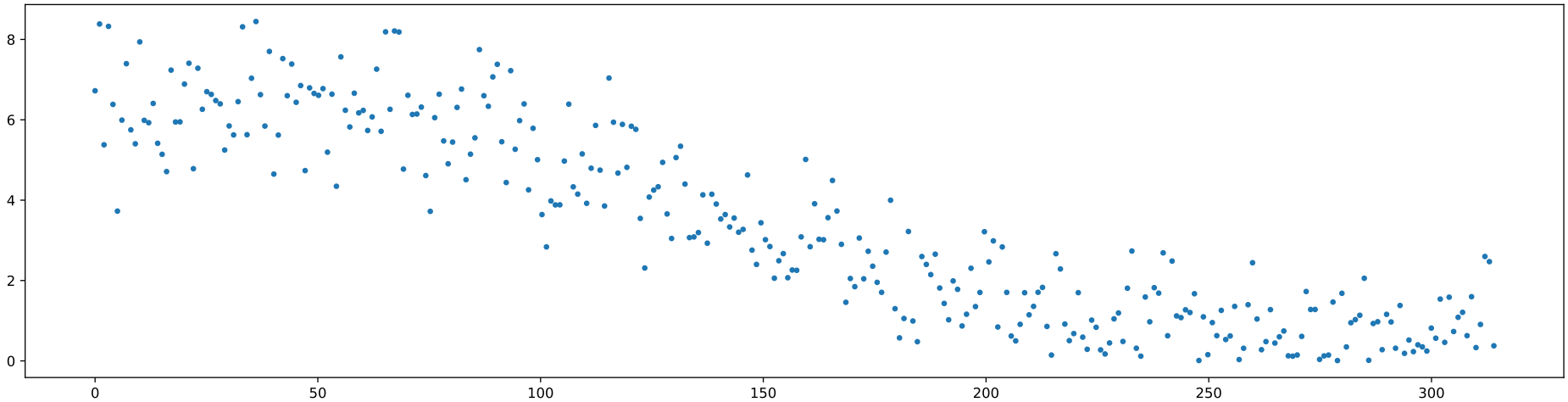


How to average-out noisy derivatives as values are coming in over time.

- Q: Weight each past derivative equally? Eg 10 steps ago vs 100 steps ago?
- A: No; more recent derivatives are closer to the current parameters. Weight recent derivatives more than older derivatives.

See a gradients as noisy time series

A blue dot is a 1d gradient (y-axis: derivative value) over iterations (x-axis: time series).

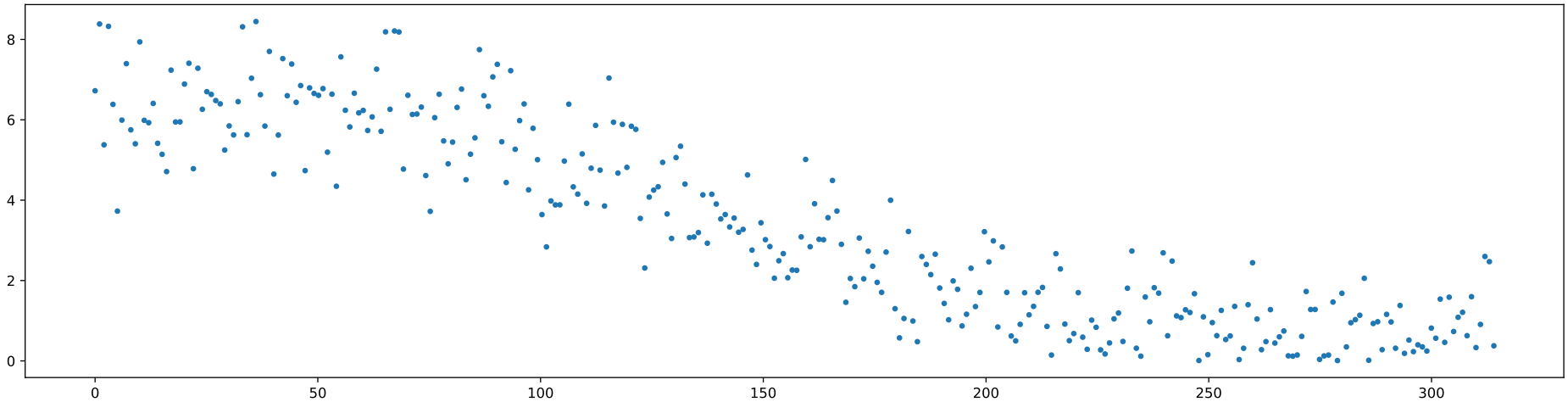


How to average-out noisy derivatives as values are coming in over time.

- Q: Weight each past derivative equally? Eg 10 steps ago vs 100 steps ago?
- A: No; more recent derivatives are closer to the current parameters. Weight recent derivatives more than older derivatives.
- Q: Let's average k recent values (moving average); practical problem?

See a gradients as noisy time series

A blue dot is a 1d gradient (y-axis: derivative value) over iterations (x-axis: time series).

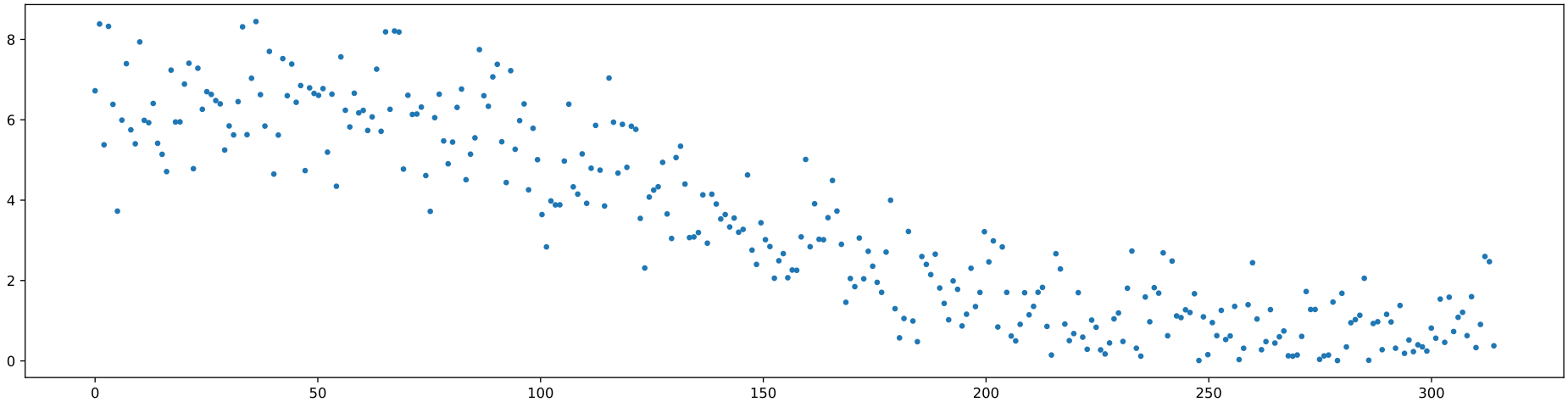


How to average-out noisy derivatives as values are coming in over time.

- Q: Weight each past derivative equally? Eg 10 steps ago vs 100 steps ago?
- A: No; more recent derivatives are closer to the current parameters. Weight recent derivatives more than older derivatives.
- Q: Let's average k recent values (moving average); practical problem?
- A: Out of memory (huge nr of params: need to store eg $1,000,000,000 \times k$)

See a gradients as noisy time series

A blue dot is a 1d gradient (y-axis: derivative value) over iterations (x-axis: time series).



How to average-out noisy derivatives as values are coming in over time.

- Q: Weight each past derivative equally? Eg 10 steps ago vs 100 steps ago?
- A: No; more recent derivatives are closer to the current parameters. Weight recent derivatives more than older derivatives.
- Q: Let's average k recent values (moving average); practical problem?
- A: Out of memory (huge nr of params: need to store eg $1,000,000,000 \times k$)

An infinite impulse response filter (IIR) to the rescue:

Recursively compute online weighted average.

Exponentially weighted moving average (EWMA)

For value y_t at time t, and $0 < \rho < 1$ and $S_{t-1} = 0$, if $t = 1$.
Exponentially weighted moving average (EWMA): $S_t = (\rho S_{t-1}) + (1 - \rho) y_t$

Exponentially weighted moving average (EWMA)

For value y_t at time t, and $0 < \rho < 1$ and $S_{t-1} = 0$, if $t = 1$.
Exponentially weighted moving average (EWMA): $S_t = (\rho S_{t-1}) + (1 - \rho) y_t$

- Q: Write it out $3\times$ for S_{100} ?

Exponentially weighted moving average (EWMA)

For value y_t at time t, and $0 < \rho < 1$ and $S_{t-1} = 0$, if $t = 1$.
Exponentially weighted moving average (EWMA): $S_t = (\rho S_{t-1}) + (1 - \rho) y_t$

- Q: Write it out $3\times$ for S_{100} ? A:

$$
\begin{aligned}
S_{100} &= \rho \quad S_{99} && + (1-\rho) y_{100} \\
S_{100} &= \rho \quad (\rho \; S_{98} && + (1-\rho) y_{99}) + (1-\rho) y_{100} \\
S_{100} &= \rho \quad (\rho \; (\rho S_{97} + (1-\rho) y_{98}) && + (1-\rho) y_{99}) + (1-\rho) y_{100}
\end{aligned}
$$

Exponentially weighted moving average (EWMA)

For value y_t at time t, and $0 < \rho < 1$ and $S_{t-1} = 0$, if $t = 1$.
Exponentially weighted moving average (EWMA): $S_t = (\rho S_{t-1}) + (1 - \rho) y_t$

- Q: Write it out $3\times$ for S_{100} ? A:

$$
\begin{aligned}
S_{100} &= \rho \quad S_{99} && + (1-\rho) y_{100} \\
S_{100} &= \rho \quad (\rho \; S_{98} && + (1-\rho) y_{99}) + (1-\rho) y_{100} \\
S_{100} &= \rho \quad (\rho \; (\rho S_{97} + (1-\rho) y_{98}) && + (1-\rho) y_{99}) + (1-\rho) y_{100}
\end{aligned}
$$

Re-order "$3\times$ for S_{100}" as an exponentially weighted sum of ys: (multiply all brackets)

Exponentially weighted moving average (EWMA)

For value y_t at time t, and $0 < \rho < 1$ and $S_{t-1} = 0$, if $t = 1$.
Exponentially weighted moving average (EWMA): $S_t = (\rho S_{t-1}) + (1 - \rho) y_t$

- Q: Write it out $3\times$ for S_{100} ? A:

$$
\begin{aligned}
S_{100} &= \rho \quad S_{99} && + (1-\rho) y_{100} \\
S_{100} &= \rho \quad (\rho \; S_{98} && + (1-\rho) y_{99}) + (1-\rho) y_{100} \\
S_{100} &= \rho \quad (\rho \; (\rho S_{97} + (1-\rho) y_{98}) && + (1-\rho) y_{99}) + (1-\rho) y_{100}
\end{aligned}
$$

Re-order "$3\times$ for S_{100}" as an exponentially weighted sum of ys: (multiply all brackets)

$$S_{100} = (1-\rho)\; y_{100} + (1-\rho)\; \rho\; y_{99} + (1-\rho)\; \rho^2\; y_{98} + (1-\rho)\; \rho^3\; y_{97} + \ldots$$

Exponentially weighted moving average (EWMA)

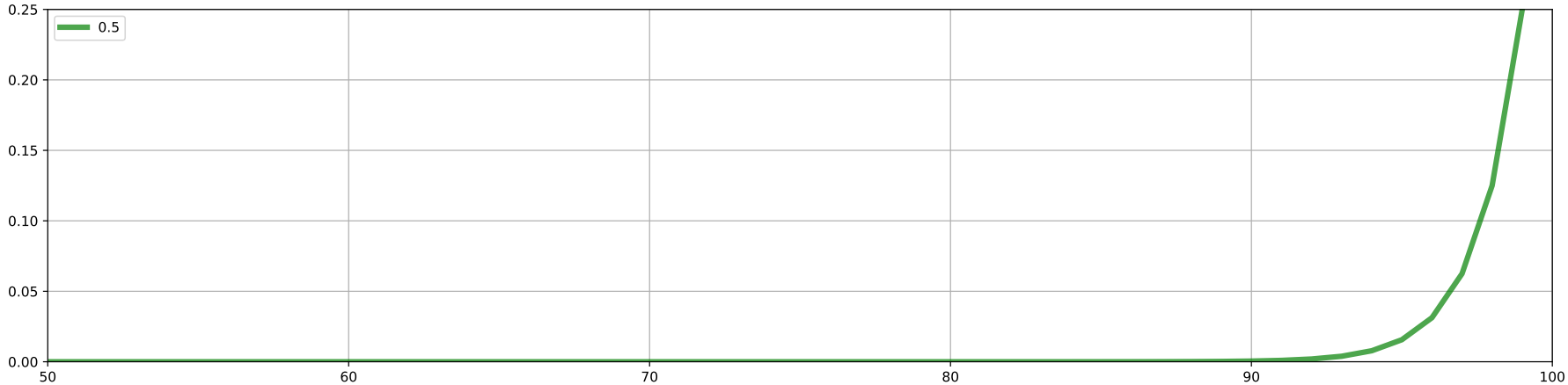
For value y_t at time t, and $0 < \rho < 1$ and $S_{t-1} = 0$, if $t = 1$.
Exponentially weighted moving average (EWMA): $S_t = (\rho S_{t-1}) + (1 - \rho) y_t$

- Q: Write it out $3\times$ for S_{100} ? A:

$$
\begin{aligned}
S_{100} &= \rho \quad S_{99} && + (1-\rho) y_{100} \\
S_{100} &= \rho \quad (\rho \; S_{98} && + (1-\rho) y_{99}) + (1-\rho) y_{100} \\
S_{100} &= \rho \quad (\rho \; (\rho S_{97} + (1-\rho) y_{98}) && + (1-\rho) y_{99}) + (1-\rho) y_{100}
\end{aligned}
$$

Re-order "$3\times$ for S_{100}" as an exponentially weighted sum of ys: (multiply all brackets)

$$S_{100} = (1-\rho)\; y_{100} + (1-\rho)\; \rho\; y_{99} + (1-\rho)\; \rho^2\; y_{98} + (1-\rho)\; \rho^3\; y_{97} + \ldots$$



Example of how the ys (on x-axis) are weighted for $\rho = 0.5$

Exponentially weighted moving average (EWMA)

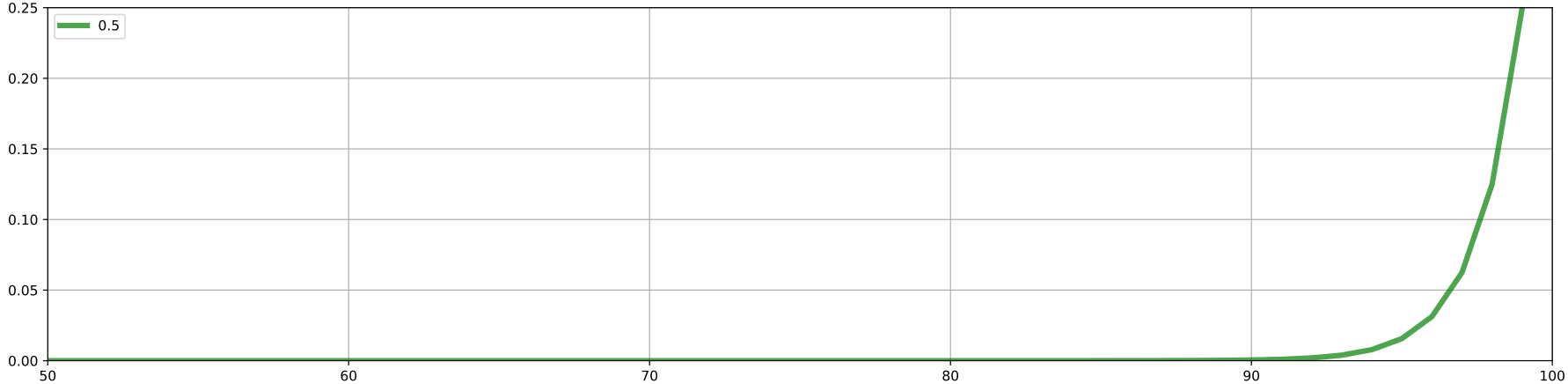
For value y_t at time t, and $0 < \rho < 1$ and $S_{t-1} = 0$, if $t = 1$.
Exponentially weighted moving average (EWMA): $S_t = (\rho S_{t-1}) + (1 - \rho) y_t$

- Q: Write it out $3\times$ for S_{100} ? A:

$$
\begin{aligned}
S_{100} &= \rho \quad S_{99} && + (1-\rho) y_{100} \\
S_{100} &= \rho \quad (\rho \; S_{98} && + (1-\rho) y_{99}) + (1-\rho) y_{100} \\
S_{100} &= \rho \quad (\rho \; (\rho S_{97} + (1-\rho) y_{98}) && + (1-\rho) y_{99}) + (1-\rho) y_{100}
\end{aligned}
$$

Re-order "$3\times$ for S_{100}" as an exponentially weighted sum of ys: (multiply all brackets)

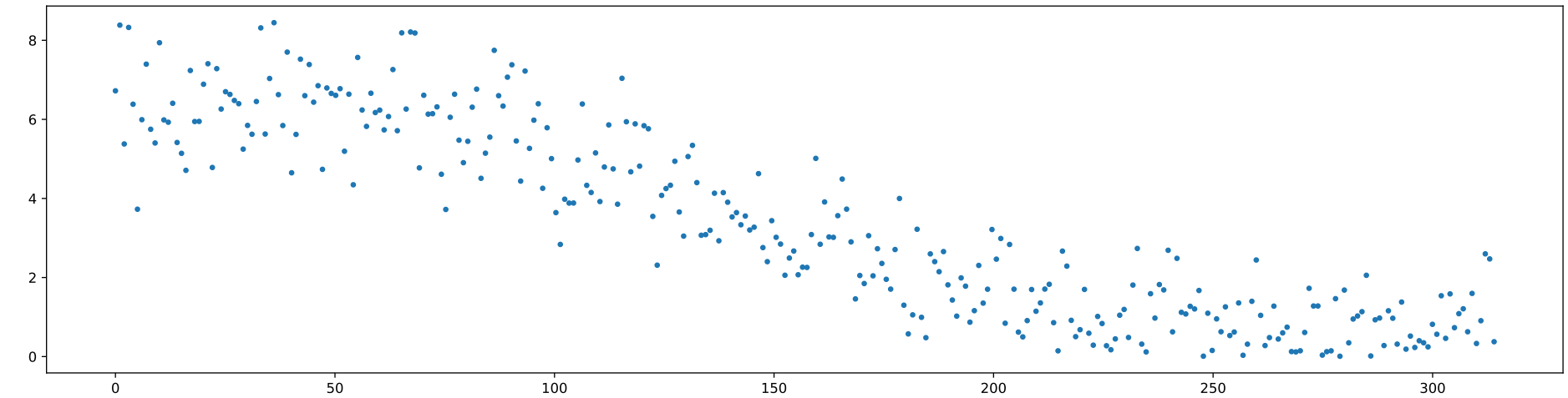
$$S_{100} = (1-\rho)\; y_{100} + (1-\rho)\; \rho\; y_{99} + (1-\rho)\; \rho^2\; y_{98} + (1-\rho)\; \rho^3\; y_{97} + \ldots$$

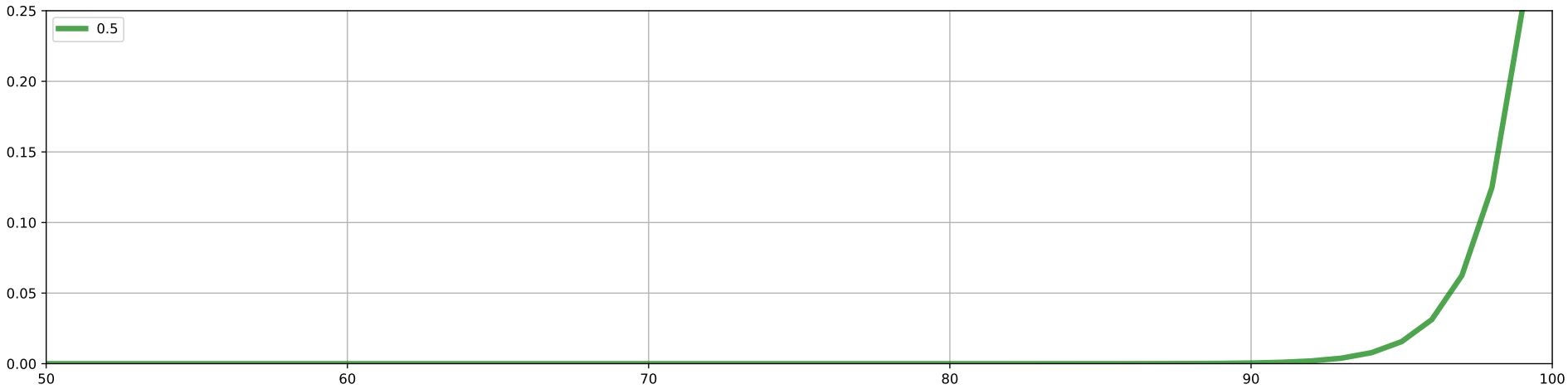


Example of how the ys (on x-axis) are weighted for $\rho = 0.5$

Q: How about $\rho = 0.9$? $\rho = 0.95$?

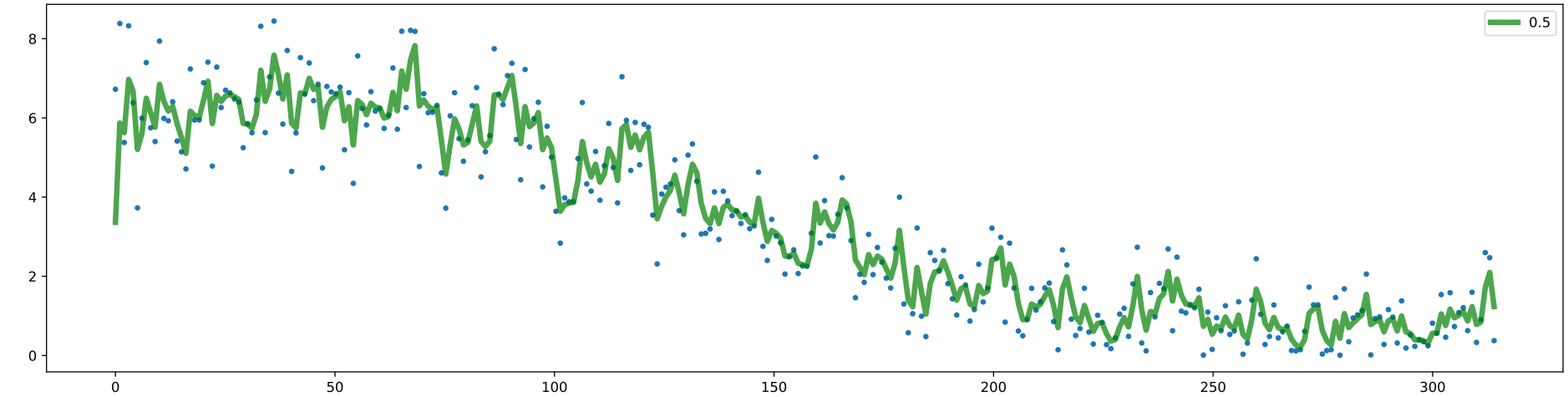
Exponentially weighted moving average (EWMA)

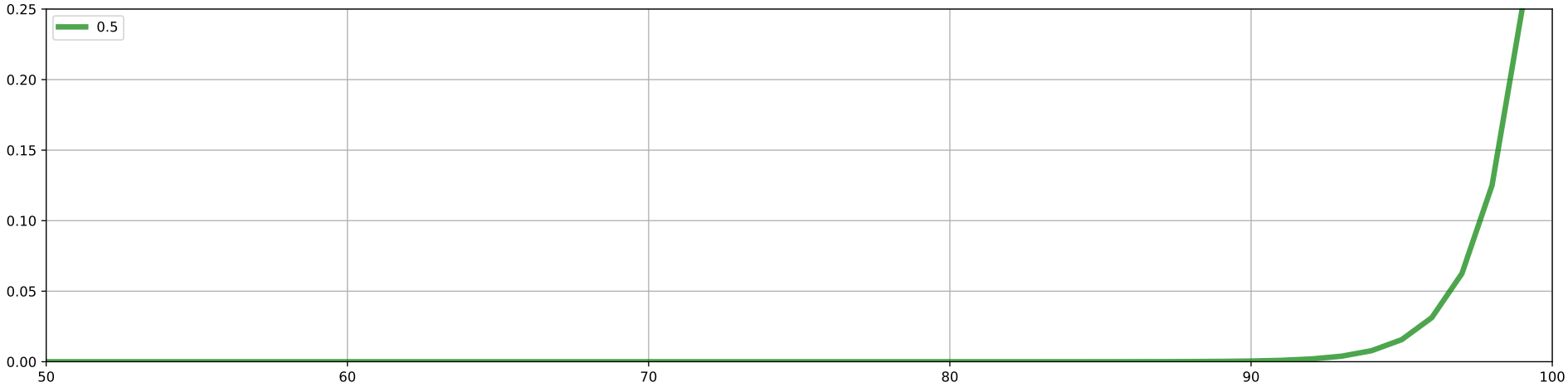




Example for $\rho = 0.5$, $\rho = 0.9$, $\rho = 0.95$

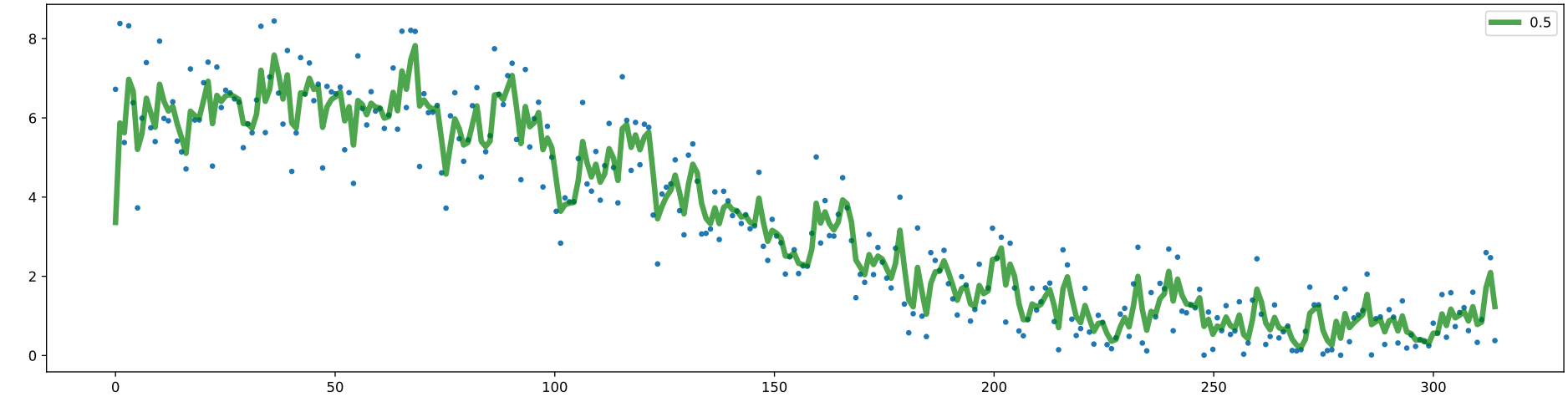
Exponentially weighted moving average (EWMA)

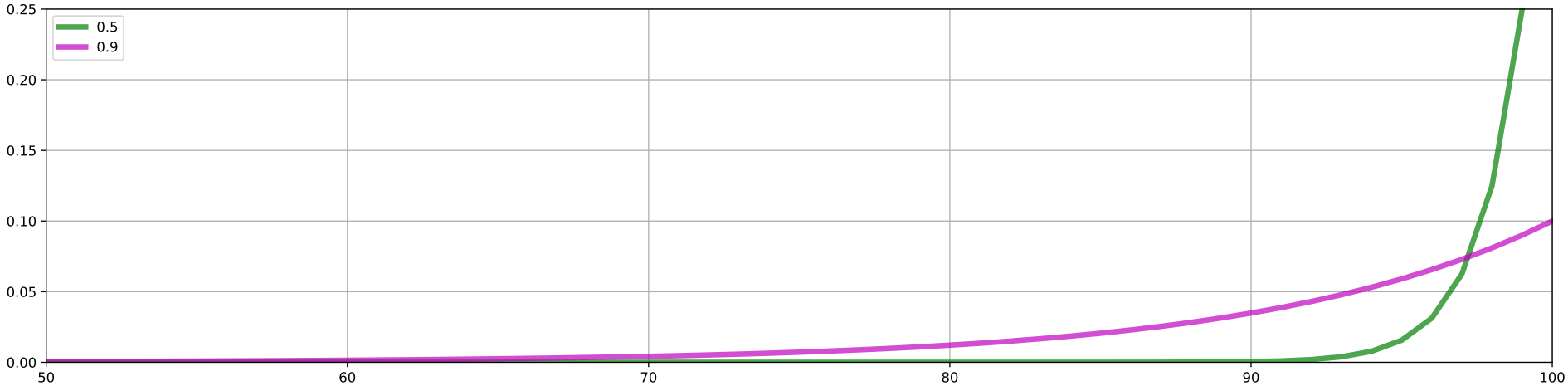




Example for $\rho = 0.5$, $\rho = 0.9$, $\rho = 0.95$

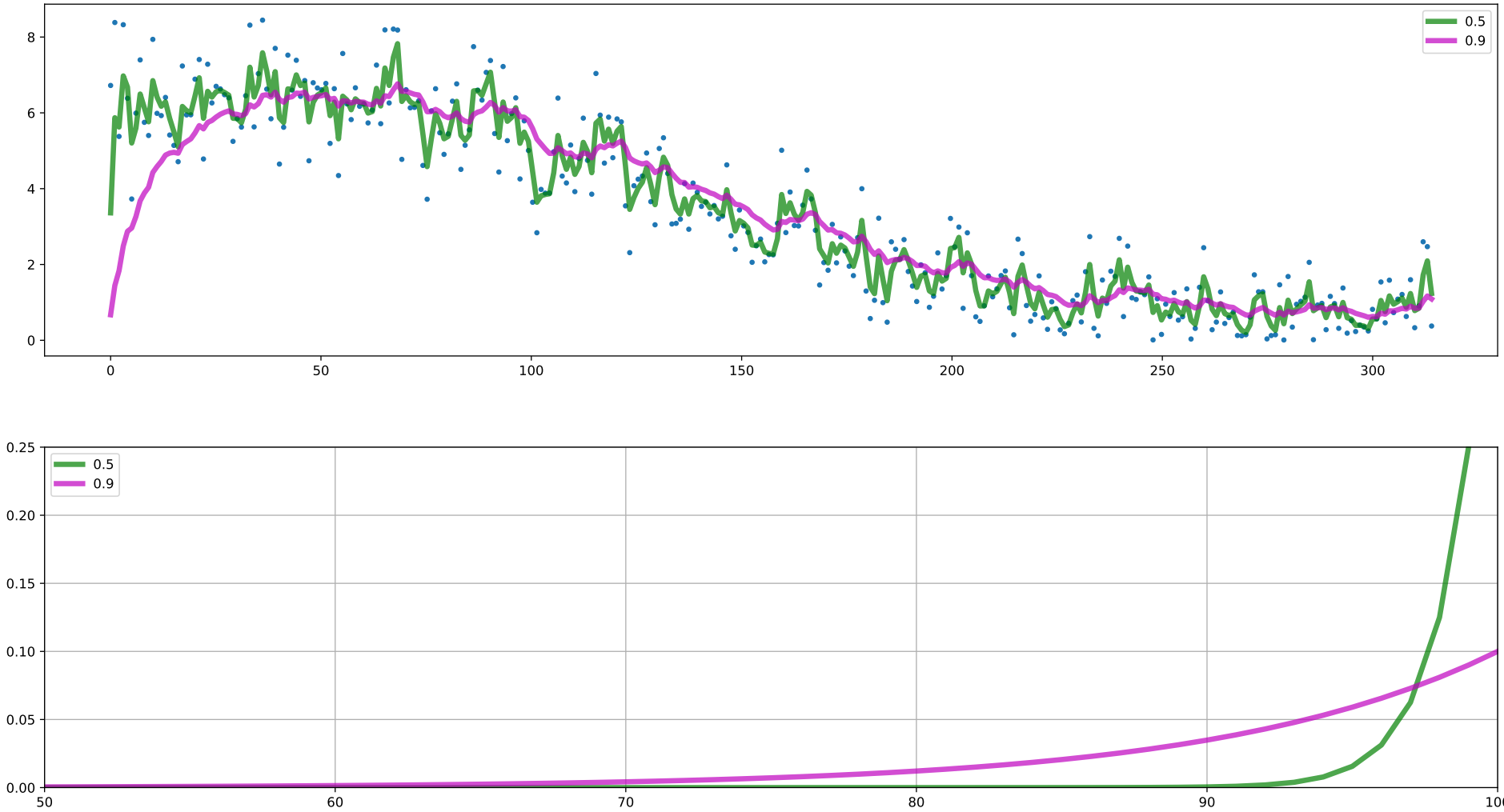
Exponentially weighted moving average (EWMA)





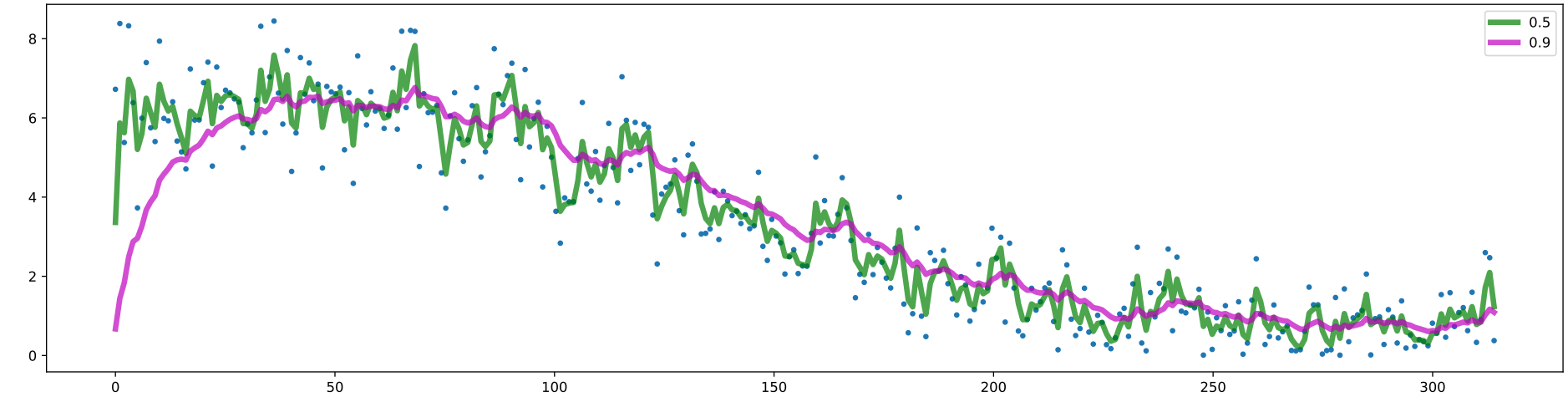
Example for $\rho = 0.5$, $\rho = 0.9$, $\rho = 0.95$

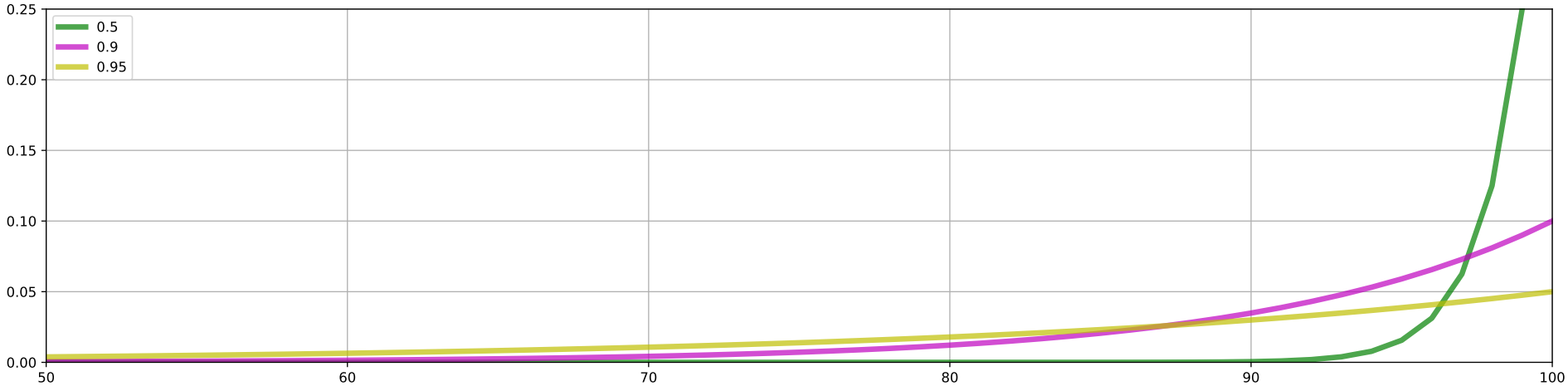
Exponentially weighted moving average (EWMA)



Example for $\rho = 0.5$, $\rho = 0.9$, $\rho = 0.95$

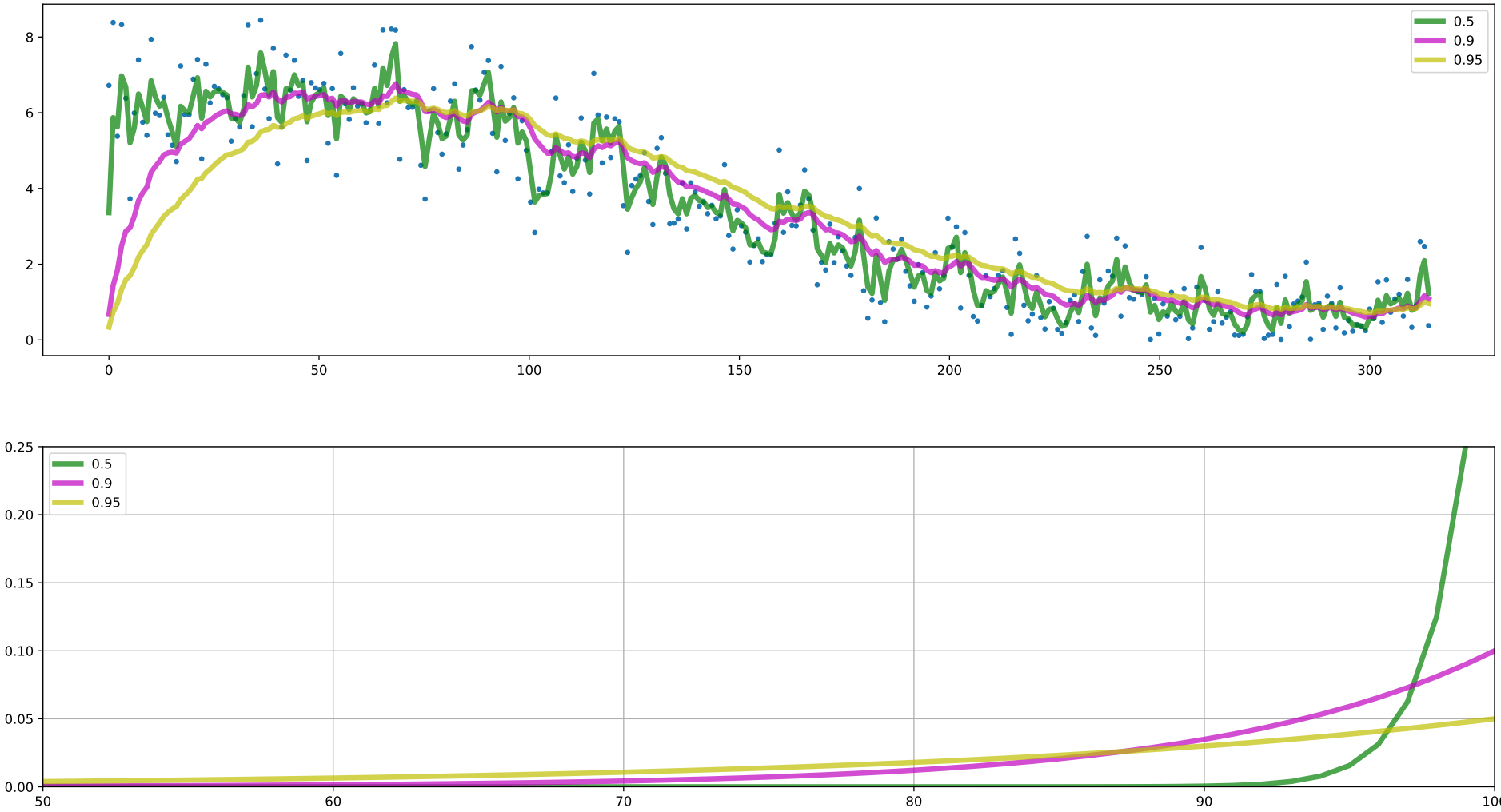
Exponentially weighted moving average (EWMA)





Example for $\rho = 0.5$, $\rho = 0.9$, $\rho = 0.95$

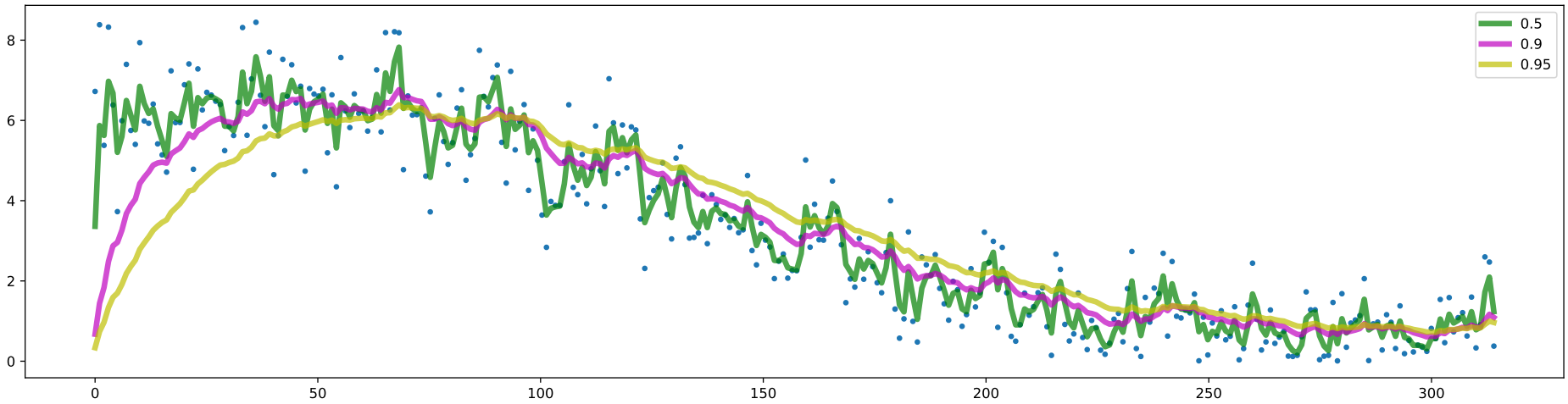
Exponentially weighted moving average (EWMA)



Example for $\rho = 0.5$, $\rho = 0.9$, $\rho = 0.95$

Conclusion exponentially weighted moving average

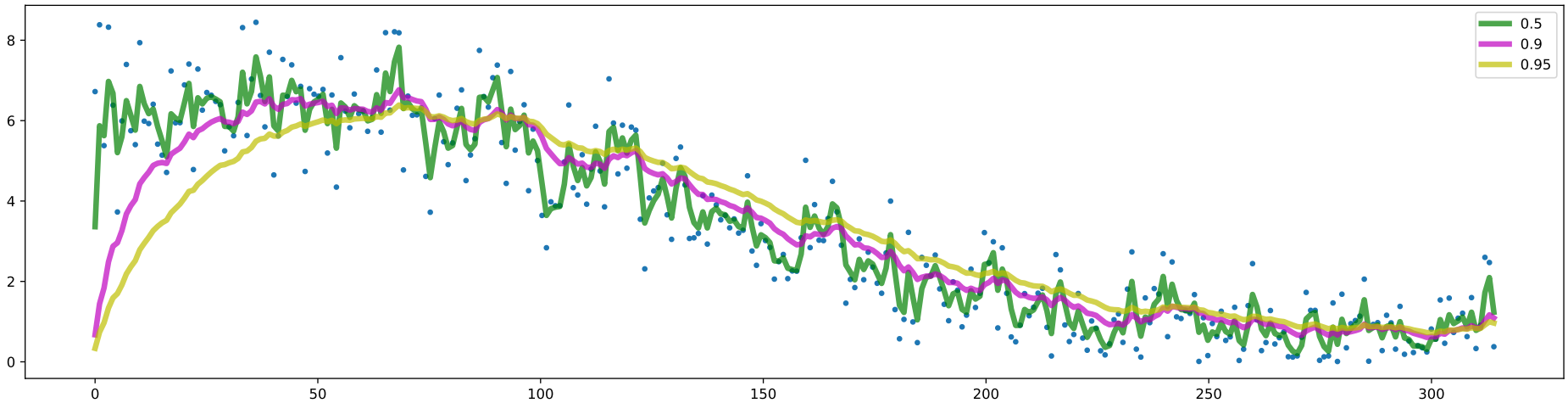
For value y_t at time t, and $0 < \rho < 1$ and $S_{t-1} = 0$, if $t = 1$. Exponentially weighted moving average (EWMA): $S_t = (\rho S_{t-1}) + (1 - \rho) y_t$



- Q: For n parameters, how many values to store to compute an average?

Conclusion exponentially weighted moving average

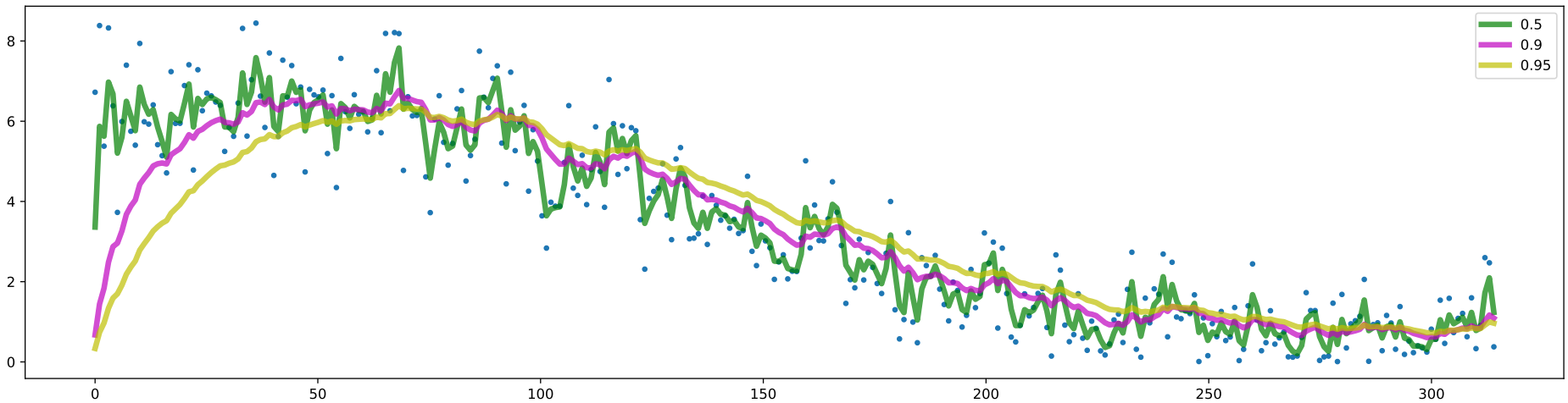
For value y_t at time t, and $0 < \rho < 1$ and $S_{t-1} = 0$, if $t = 1$. Exponentially weighted moving average (EWMA): $S_t = (\rho S_{t-1}) + (1 - \rho) y_t$



- Q: For n parameters, how many values to store to compute an average?
- A: $2n$ (n extra values to store; one for each parameter)

Conclusion exponentially weighted moving average

For value y_t at time t, and $0 < \rho < 1$ and $S_{t-1} = 0$, if $t = 1$. Exponentially weighted moving average (EWMA): $S_t = (\rho S_{t-1}) + (1 - \rho) y_t$

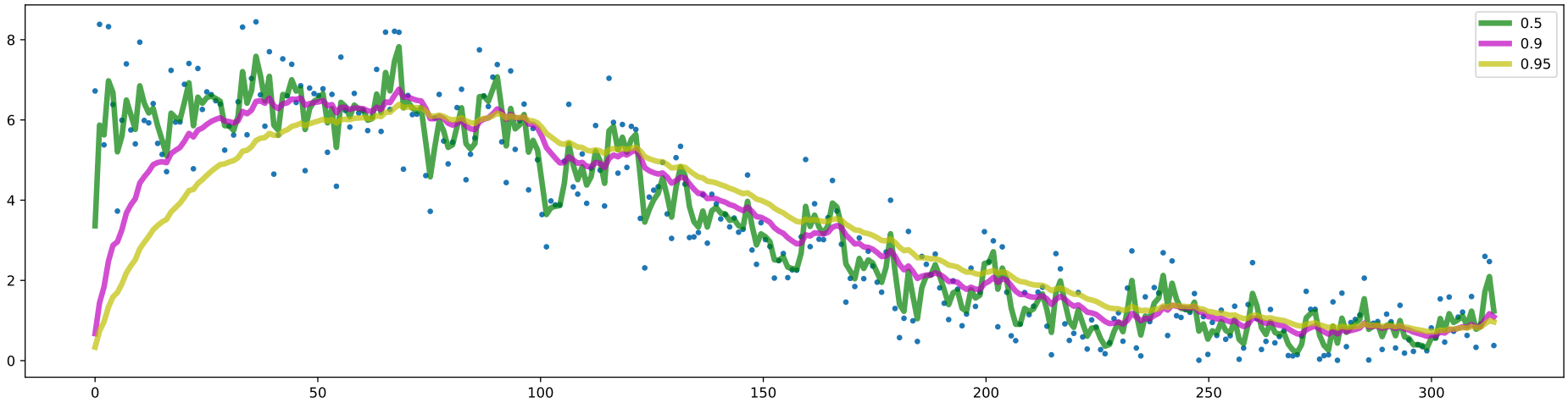


- Q: For n parameters, how many values to store to compute an average?
- A: $2n$ (n extra values to store; one for each parameter)

EWMA: Memory efficient approach to compute summary statistics online

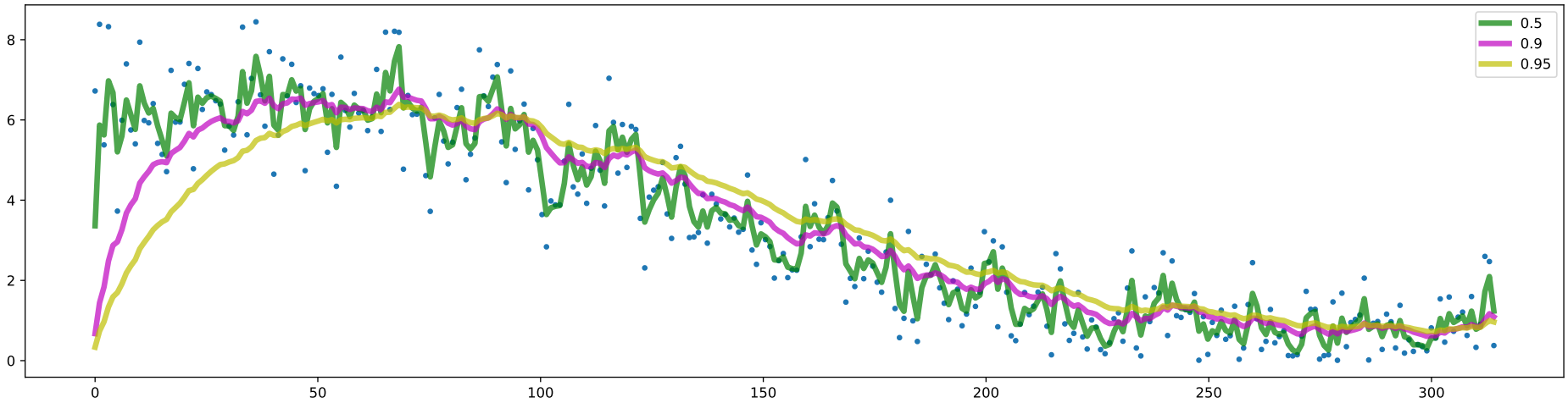
Questions?

Bias



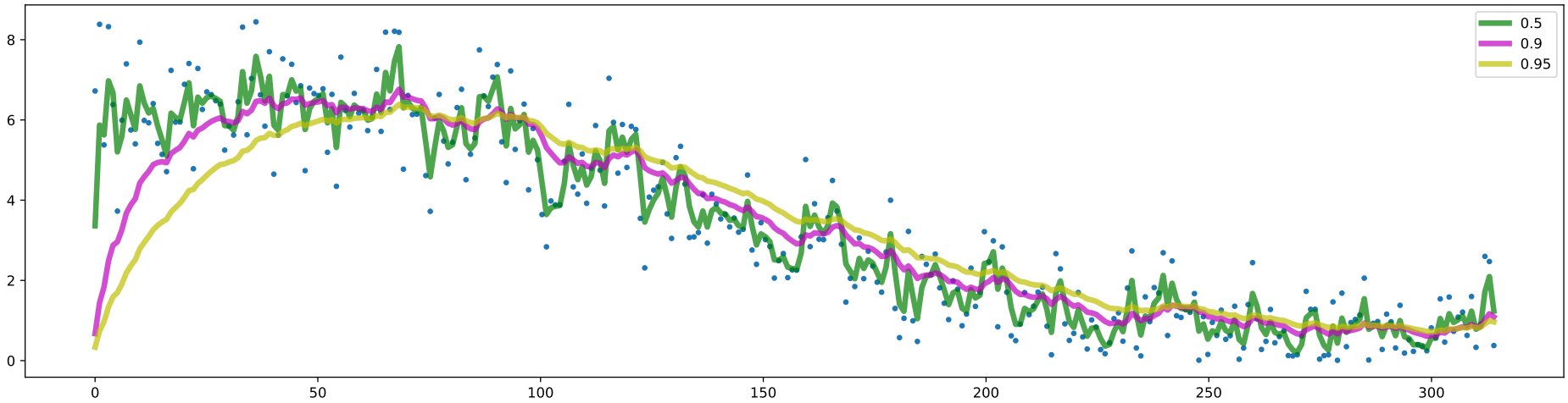
- Q: What do you notice about these curves?

Bias



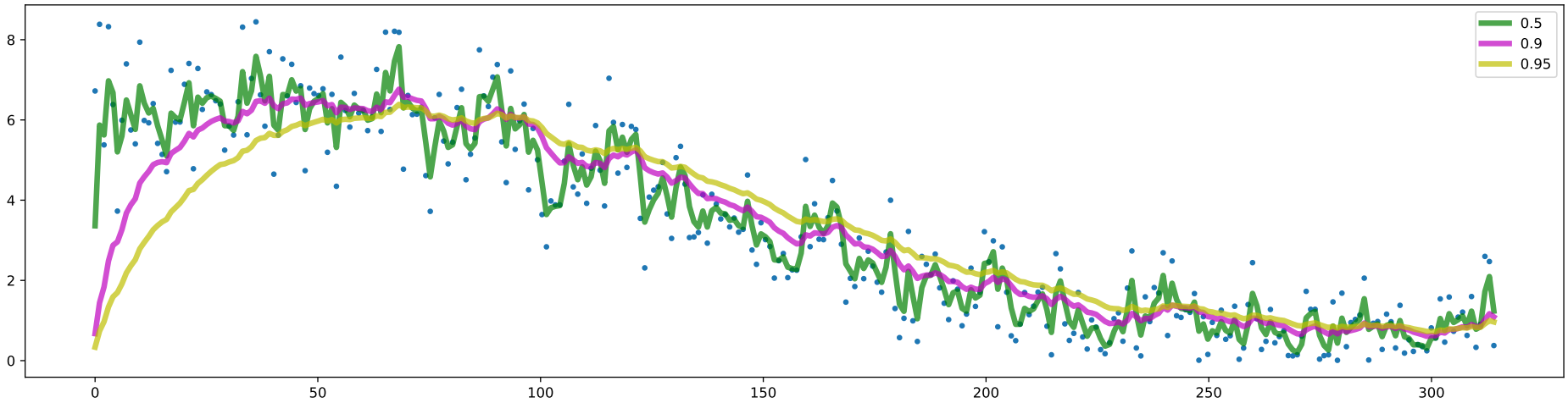
- Q: What do you notice about these curves?
- A: The larger ρ, the later it 'catches up'.

Bias



- Q: What do you notice about these curves?
- A: The larger ρ, the later it 'catches up'.
- Q: Why is that?

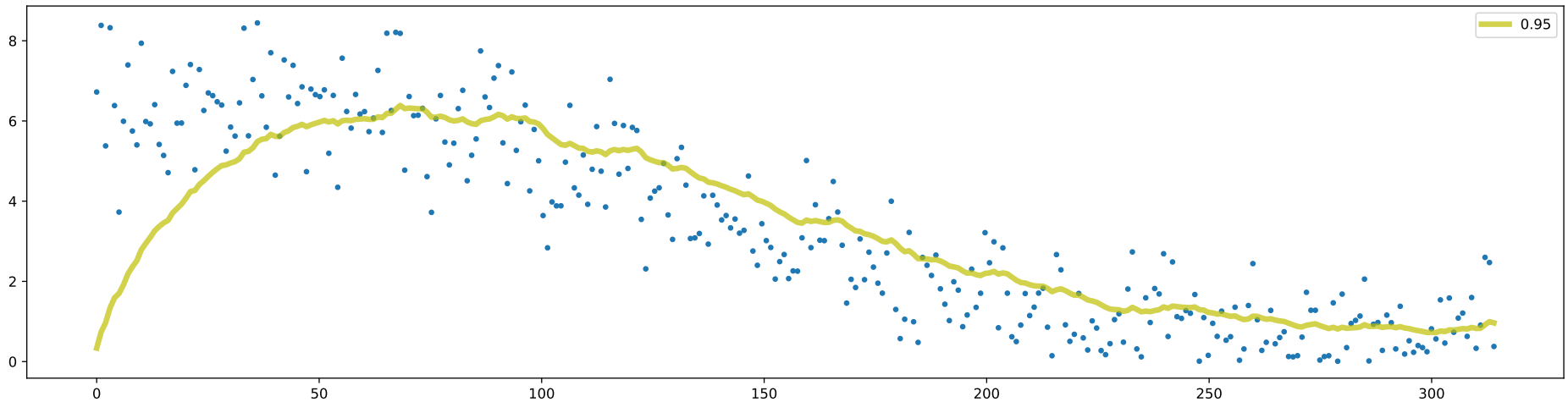
Bias



- Q: What do you notice about these curves?
- A: The larger ρ, the later it 'catches up'.
- Q: Why is that?
- A: Border effect: Unknown what happened before time.

Bias correction for $S_t = (\rho S_{t-1}) + (1-\rho)y_t$

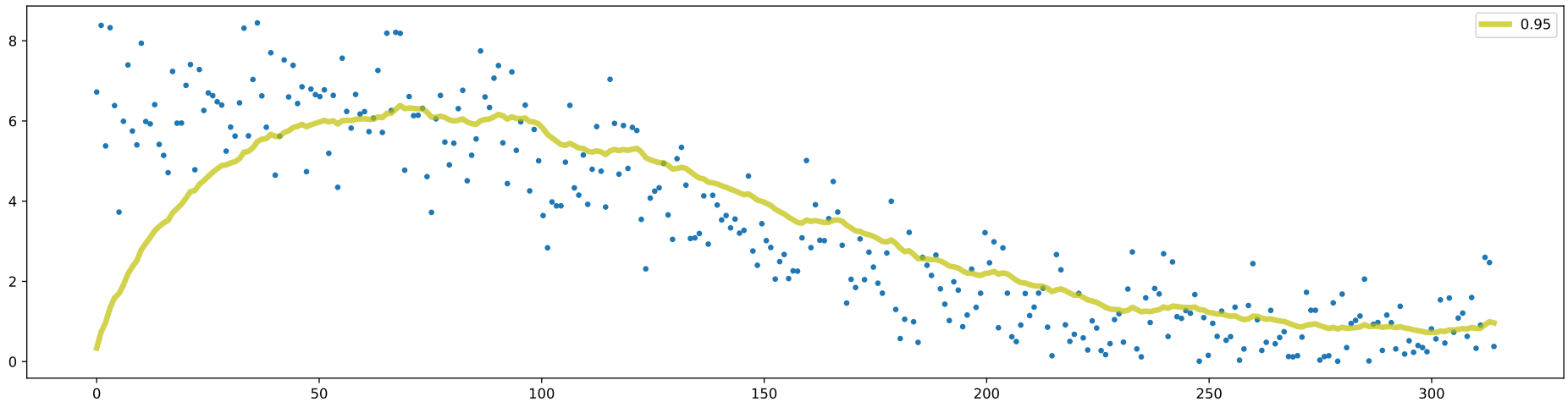
Q: Compute S_1 and S_2 for $\rho = 0.95$?



Bias correction for $S_t = (\rho S_{t-1}) + (1-\rho)y_t$

Q: Compute S_1 and S_2 for $\rho = 0.95$? A:

- $S_1 = \rho\ S_0 + (1-\rho)y_1 = 0 + 0.05y_1$
- $S_2 = \rho\ S_1 + (1-\rho)y_2 = 0.95 \times 0.05y_1 + 0.05y_2 = 0.0475y_1 + 0.05y_2$

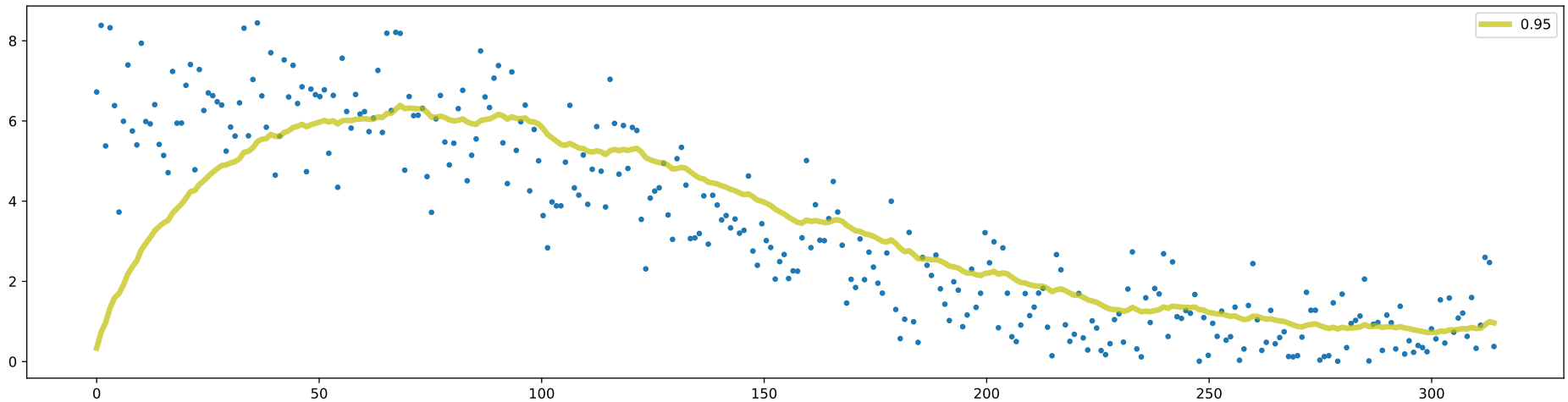


Bias correction for $S_t = (\rho S_{t-1}) + (1-\rho)y_t$

Q: Compute S_1 and S_2 for $\rho = 0.95$? A:

- $S_1 = \rho\ S_0 + (1-\rho)y_1 = 0 + 0.05y_1$
- $S_2 = \rho\ S_1 + (1-\rho)y_2 = 0.95 \times 0.05y_1 + 0.05y_2 = 0.0475y_1 + 0.05y_2$

Indeed, very low values at the beginning.



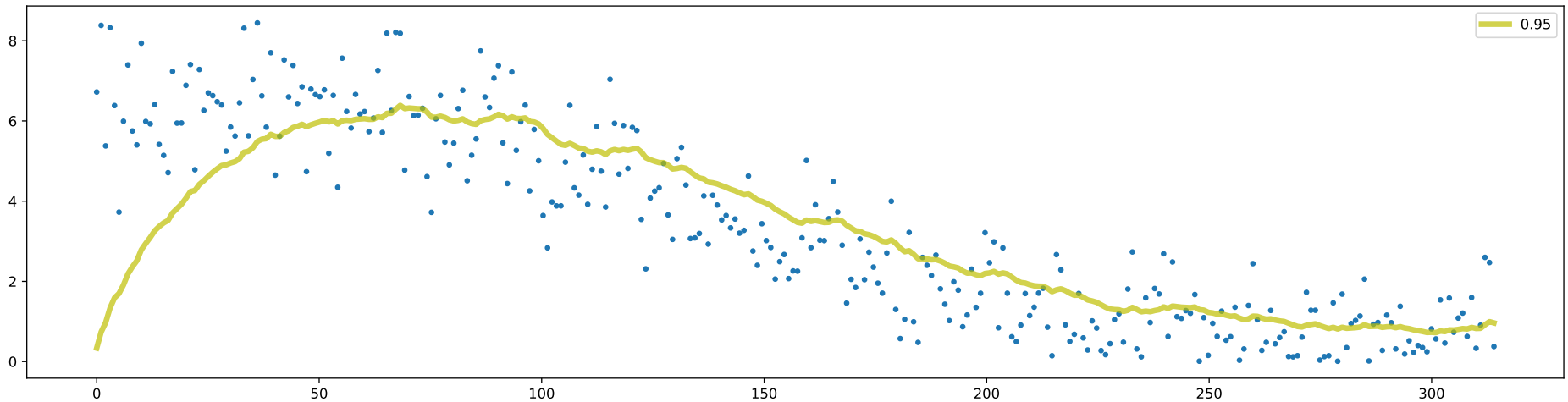
Bias correction for $S_t = (\rho S_{t-1}) + (1-\rho)y_t$

Q: Compute S_1 and S_2 for $\rho = 0.95$? A:

- $S_1 = \rho\ S_0 + (1-\rho)y_1 = 0 + 0.05y_1$
- $S_2 = \rho\ S_1 + (1-\rho)y_2 = 0.95 \times 0.05y_1 + 0.05y_2 = 0.0475y_1 + 0.05y_2$

Indeed, very low values at the beginning.

Bias correction: $\hat{S}_t = \frac{S_t}{1-\rho^t}$



Bias correction for $S_t = (\rho S_{t-1}) + (1-\rho)y_t$

Q: Compute S_1 and S_2 for $\rho = 0.95$? A:

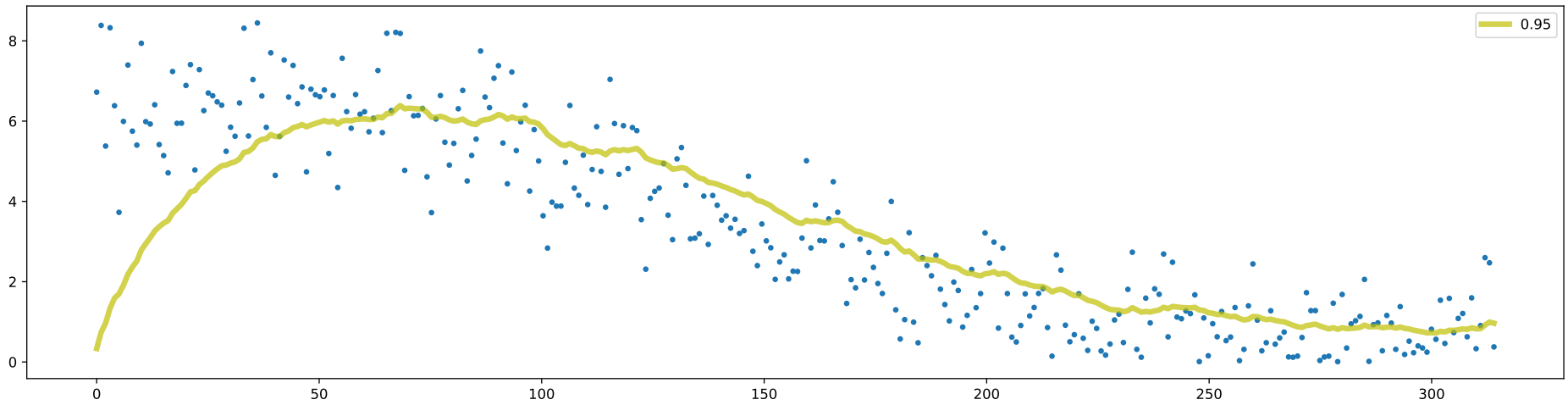
- $S_1 = \rho\ S_0 + (1-\rho)y_1 = 0 + 0.05y_1$
- $S_2 = \rho\ S_1 + (1-\rho)y_2 = 0.95 \times 0.05y_1 + 0.05y_2 = 0.0475y_1 + 0.05y_2$

Indeed, very low values at the beginning.

$$\text{Bias correction: } \hat{S}_t = \frac{S_t}{1-\rho^t}$$

For $t = 2$, the value $1 - \rho^t = 0.0975$, so: $\frac{0.0475y_1 + 0.05y_2}{0.0975}$

Q: Notice?



Bias correction for $S_t = (\rho S_{t-1}) + (1-\rho)y_t$

Q: Compute S_1 and S_2 for $\rho = 0.95$? A:

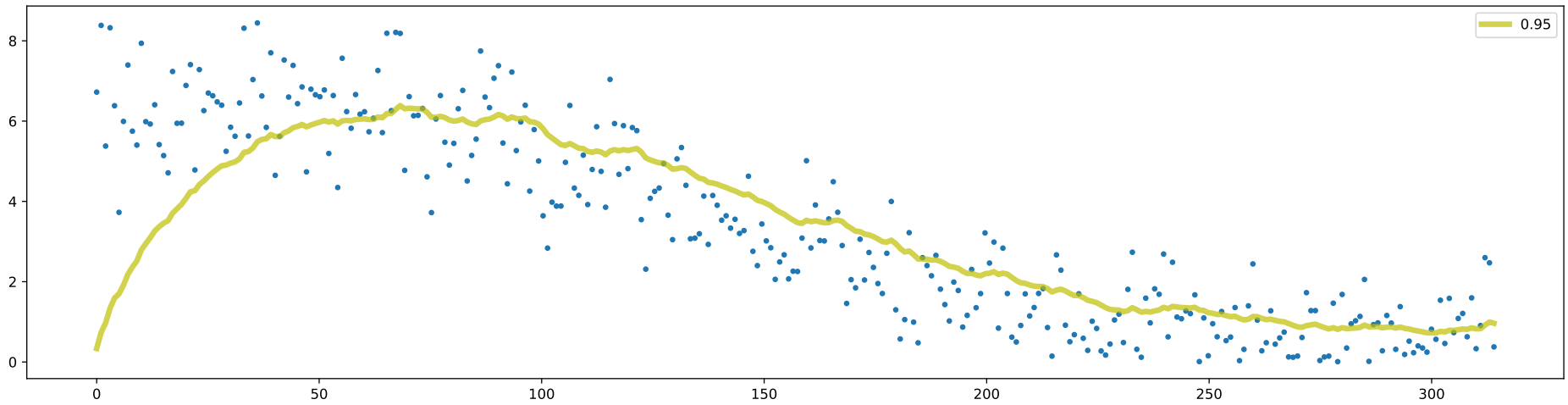
- $S_1 = \rho\ S_0 + (1-\rho)y_1 = 0 + 0.05y_1$
- $S_2 = \rho\ S_1 + (1-\rho)y_2 = 0.95 \times 0.05y_1 + 0.05y_2 = 0.0475y_1 + 0.05y_2$

Indeed, very low values at the beginning.

$$\text{Bias correction: } \hat{S}_t = \frac{S_t}{1-\rho^t}$$

For $t = 2$, the value $1 - \rho^t = 0.0975$, so: $\frac{0.0475y_1 + 0.05y_2}{0.0975}$

Q: Notice? A: Exactly normalizes the average $(0.0475 + 0.05 = 0.0975)$



Bias correction for $S_t = (\rho S_{t-1}) + (1-\rho)y_t$

Q: Compute S_1 and S_2 for $\rho = 0.95$? A:

- $S_1 = \rho\ S_0 + (1-\rho)y_1 = 0 + 0.05y_1$
- $S_2 = \rho\ S_1 + (1-\rho)y_2 = 0.95 \times 0.05y_1 + 0.05y_2 = 0.0475y_1 + 0.05y_2$

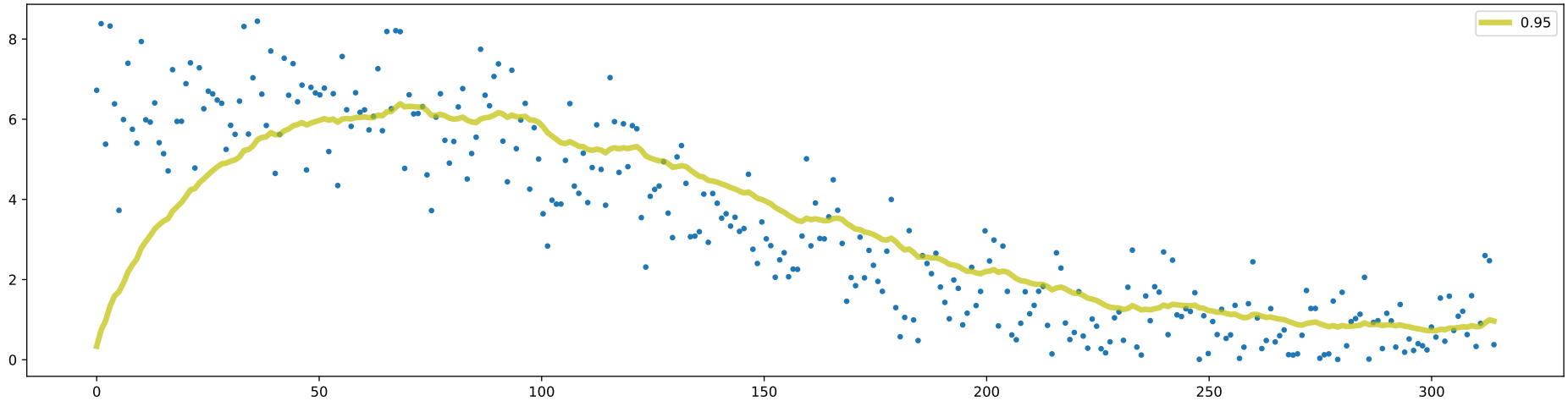
Indeed, very low values at the beginning.

$$\text{Bias correction: } \hat{S}_t = \frac{S_t}{1-\rho^t}$$

For $t = 2$, the value $1 - \rho^t = 0.0975$, so: $\frac{0.0475y_1 + 0.05y_2}{0.0975}$

Q: Notice? A: Exactly normalizes the average $(0.0475 + 0.05 = 0.0975)$

Q: What happens to the bias for large t?



Bias correction for $S_t = (\rho S_{t-1}) + (1-\rho)y_t$

Q: Compute S_1 and S_2 for $\rho = 0.95$? A:

- $S_1 = \rho\ S_0 + (1-\rho)y_1 = 0 + 0.05y_1$
- $S_2 = \rho\ S_1 + (1-\rho)y_2 = 0.95 \times 0.05y_1 + 0.05y_2 = 0.0475y_1 + 0.05y_2$

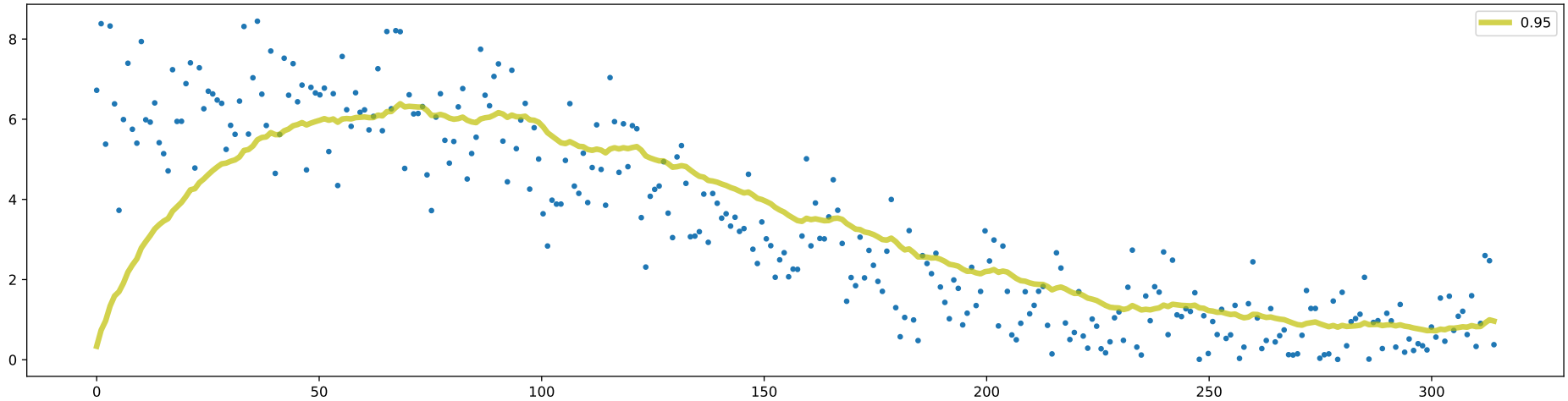
Indeed, very low values at the beginning.

$$\text{Bias correction: } \hat{S}_t = \frac{S_t}{1-\rho^t}$$

For $t = 2$, the value $1 - \rho^t = 0.0975$, so: $\frac{0.0475y_1 + 0.05y_2}{0.0975}$

Q: Notice? A: Exactly normalizes the average $(0.0475 + 0.05 = 0.0975)$

Q: What happens to the bias for large t? $1 - \rho^t \approx 1$, so $\hat{S}_t \approx S_t$



Bias correction for $S_t = (\rho S_{t-1}) + (1-\rho)y_t$

Q: Compute S_1 and S_2 for $\rho = 0.95$? A:

- $S_1 = \rho\ S_0 + (1-\rho)y_1 = 0 + 0.05y_1$
- $S_2 = \rho\ S_1 + (1-\rho)y_2 = 0.95 \times 0.05y_1 + 0.05y_2 = 0.0475y_1 + 0.05y_2$

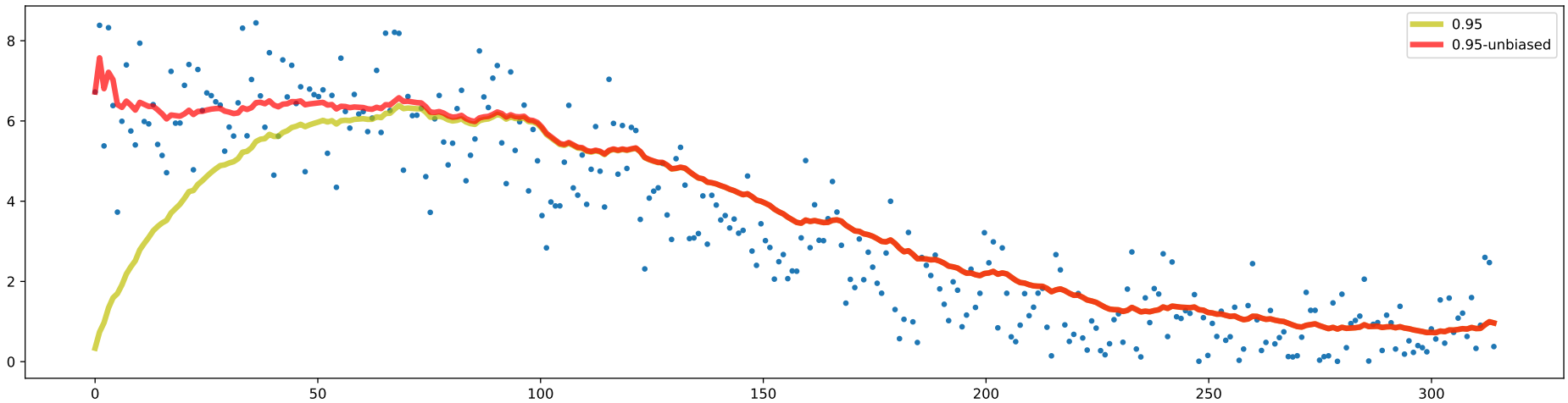
Indeed, very low values at the beginning.

$$\text{Bias correction: } \hat{S}_t = \frac{S_t}{1-\rho^t}$$

For $t = 2$, the value $1 - \rho^t = 0.0975$, so: $\frac{0.0475y_1 + 0.05y_2}{0.0975}$

Q: Notice? A: Exactly normalizes the average $(0.0475 + 0.05 = 0.0975)$

Q: What happens to the bias for large t? $1 - \rho^t \approx 1$, so $\hat{S}_t \approx S_t$



Bias correction for $S_t = (\rho S_{t-1}) + (1-\rho)y_t$

Q: Compute S_1 and S_2 for $\rho = 0.95$? A:

- $S_1 = \rho\ S_0 + (1-\rho)y_1 = 0 + 0.05y_1$
- $S_2 = \rho\ S_1 + (1-\rho)y_2 = 0.95 \times 0.05y_1 + 0.05y_2 = 0.0475y_1 + 0.05y_2$

Indeed, very low values at the beginning.

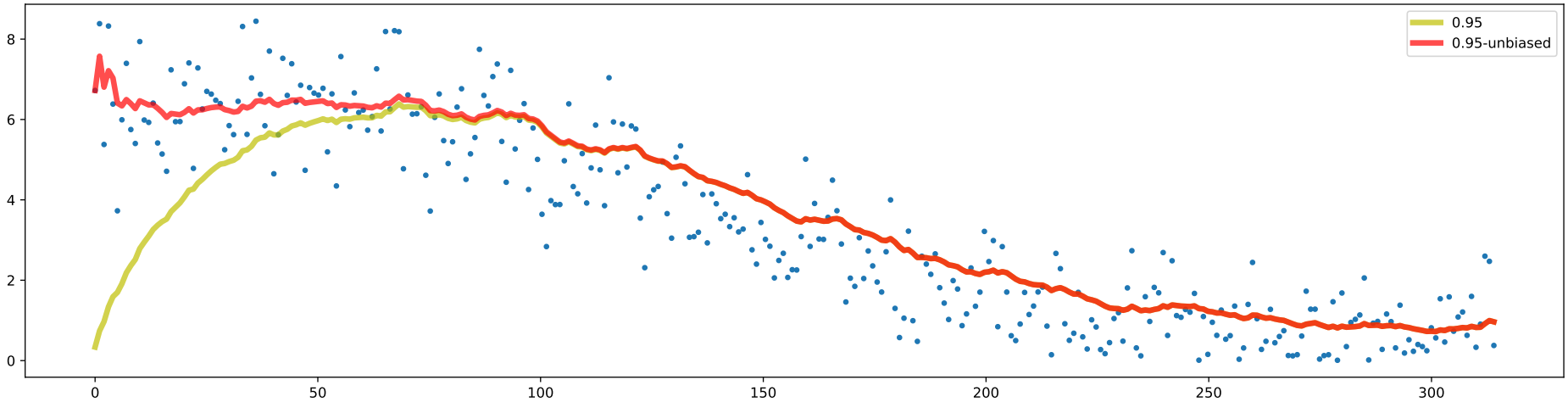
$$\text{Bias correction: } \hat{S}_t = \frac{S_t}{1-\rho^t}$$

For $t = 2$, the value $1 - \rho^t = 0.0975$, so: $\frac{0.0475y_1 + 0.05y_2}{0.0975}$

Q: Notice? A: Exactly normalizes the average $(0.0475 + 0.05 = 0.0975)$

Q: What happens to the bias for large t? $1 - \rho^t \approx 1$, so $\hat{S}_t \approx S_t$

Q: Why less smooth at beginning?



Bias correction for $S_t = (\rho S_{t-1}) + (1-\rho)y_t$

Q: Compute S_1 and S_2 for $\rho = 0.95$? A:

- $S_1 = \rho\ S_0 + (1-\rho)y_1 = 0 + 0.05y_1$
- $S_2 = \rho\ S_1 + (1-\rho)y_2 = 0.95 \times 0.05y_1 + 0.05y_2 = 0.0475y_1 + 0.05y_2$

Indeed, very low values at the beginning.

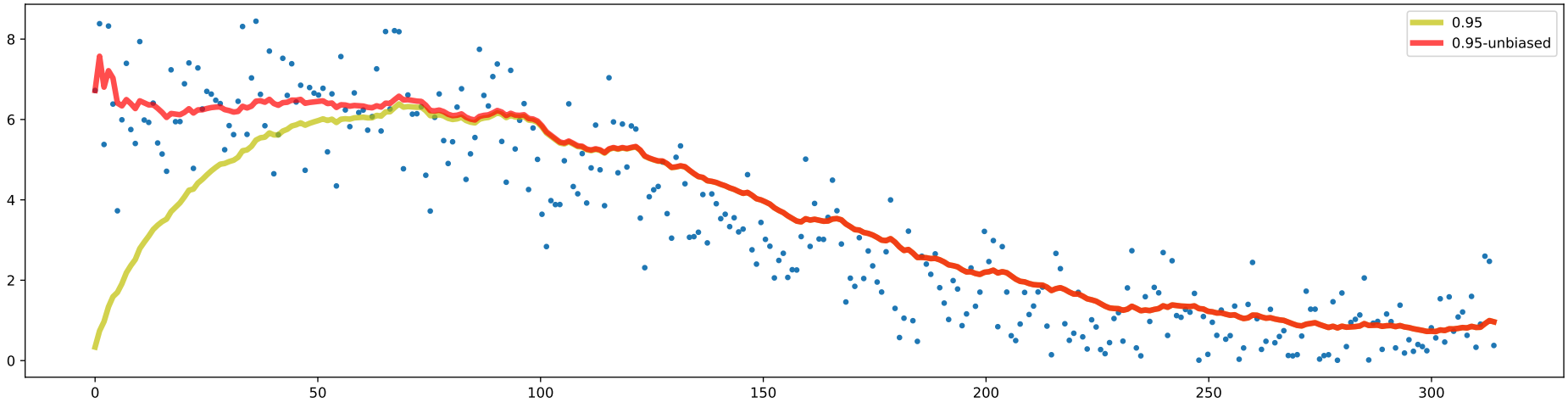
$$\text{Bias correction: } \hat{S}_t = \frac{S_t}{1-\rho^t}$$

For $t = 2$, the value $1 - \rho^t = 0.0975$, so: $\frac{0.0475y_1 + 0.05y_2}{0.0975}$

Q: Notice? A: Exactly normalizes the average $(0.0475 + 0.05 = 0.0975)$

Q: What happens to the bias for large t? $1 - \rho^t \approx 1$, so $\hat{S}_t \approx S_t$

Q: Why less smooth at beginning? A: Small sample size average



Questions?

Questions?

Useful algorithms:

- Momentum: larger LR if past partial derivative directions are similar.
- RMSprop: smaller LR if past partial derivative magnitudes are large.
- Adam: combines momentum and RMSprop.

All these algorithms make use of past gradient update statistics.

Questions?

Useful algorithms:

- Momentum: larger LR if past partial derivative directions are similar.
- RMSprop: smaller LR if past partial derivative magnitudes are large.
- Adam: combines momentum and RMSprop.

All these algorithms make use of past gradient update statistics.

Now can efficiently keep track of such statistics.

Questions?

Useful algorithms:

- Momentum: larger LR if past partial derivative directions are similar.
- RMSprop: smaller LR if past partial derivative magnitudes are large.
- Adam: combines momentum and RMSprop.

All these algorithms make use of past gradient update statistics.

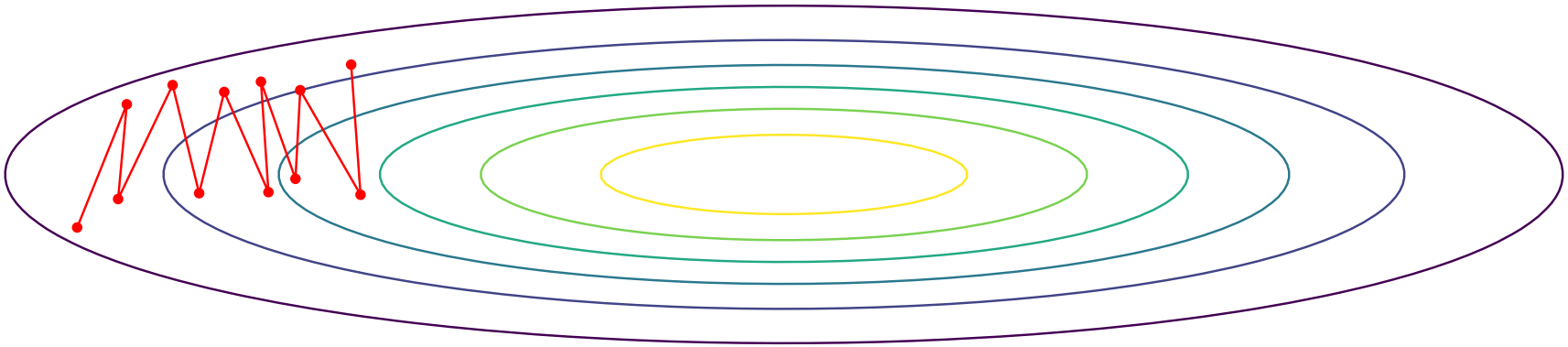
Now can efficiently keep track of such statistics.

Next slides: lets use them for gradient updates.

Stochastic Gradient Descent with momentum

Chapter 8.3.2

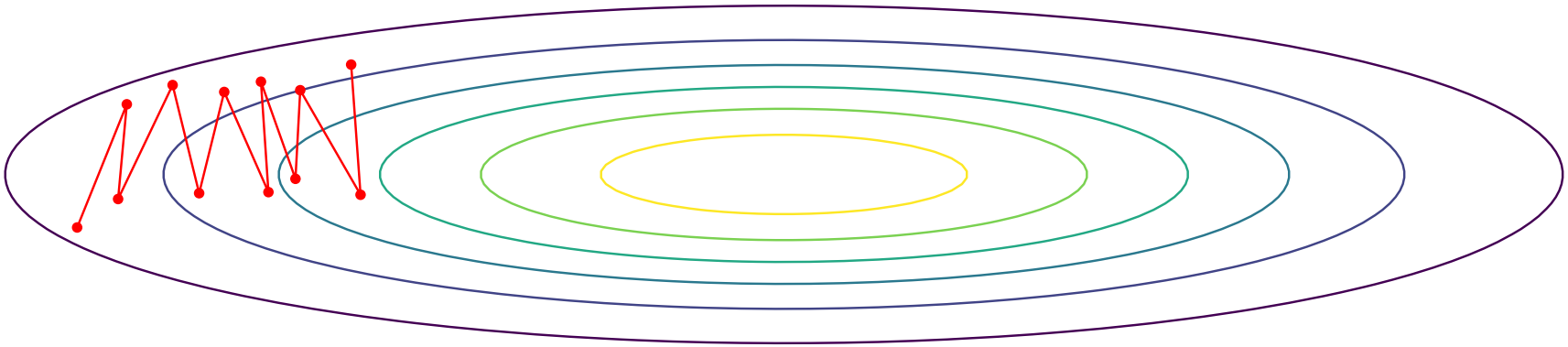
Momentum smooths the average of noisy gradients with EWMA



Stochastic Gradient Descent with momentum

Chapter 8.3.2

Momentum smooths the average of noisy gradients with EWMA



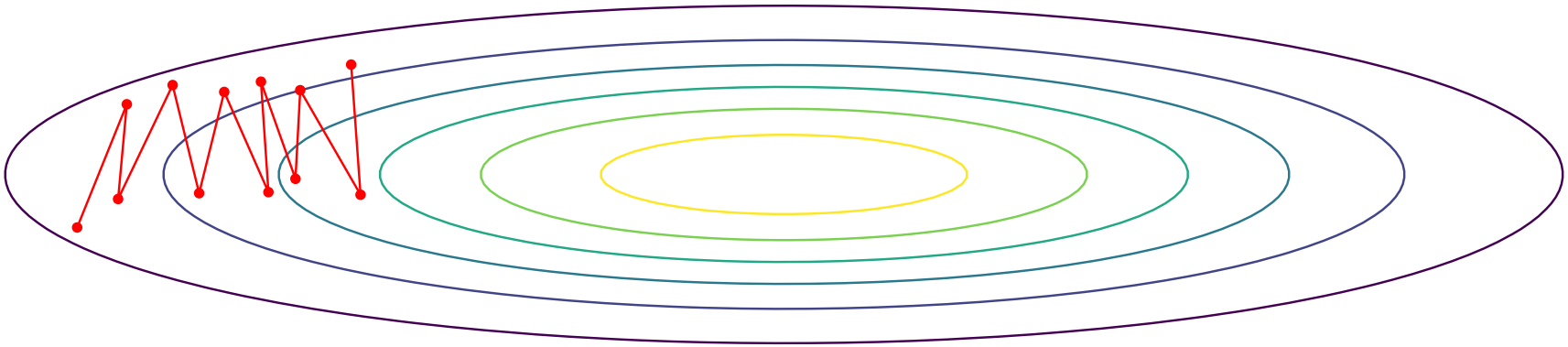
Hyper-parameters: EWMA: ρ; LR: ϵ; with ∇_θ for a mini-batch in iteration i:

- $v_i = \rho v_{i-1} + (1-\rho)\nabla_\theta$
- $\theta^* = \theta - \epsilon v_i$

Stochastic Gradient Descent with momentum

Chapter 8.3.2

Momentum smooths the average of noisy gradients with EWMA



Hyper-parameters: EWMA: ρ; LR: ϵ; with ∇_θ for a mini-batch in iteration i:

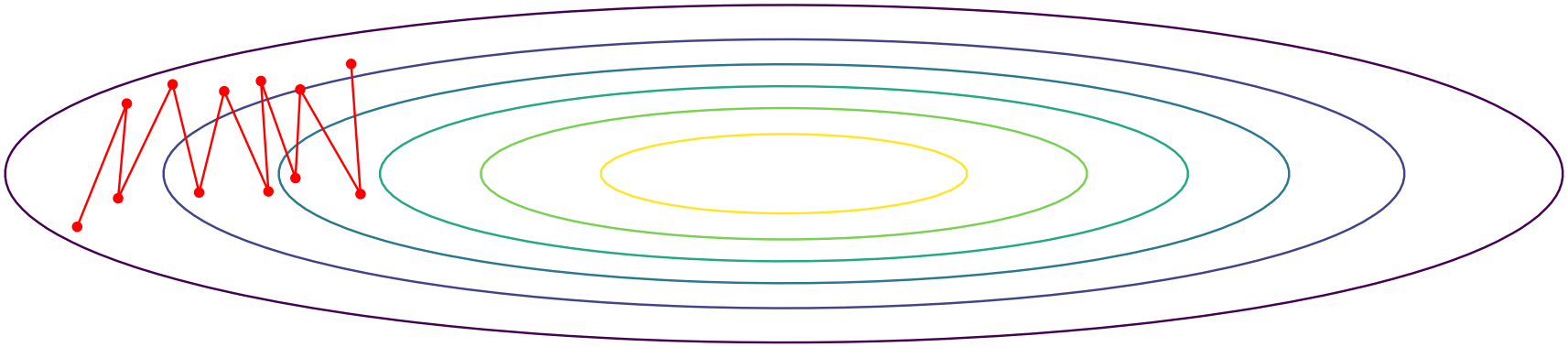
- $v_i = \rho v_{i-1} + (1-\rho)\nabla_\theta$
- $\theta^* = \theta - \epsilon v_i$

Sometimes (book) written as $v_i = \rho v_{i-1} + \nabla_\theta$
Often implemented without bias correction.
Default setting $\rho = 0.9$

Stochastic Gradient Descent with RMSprop

Chapter 8.5.2

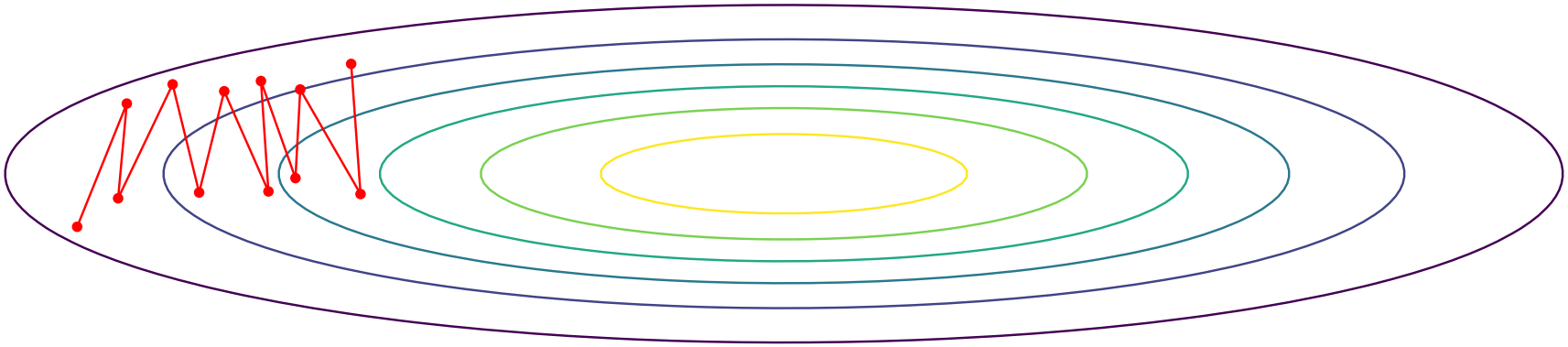
RMSprop smooths the step size of noisy gradients with EWMA



Stochastic Gradient Descent with RMSprop

Chapter 8.5.2

RMSprop smooths the step size of noisy gradients with EWMA



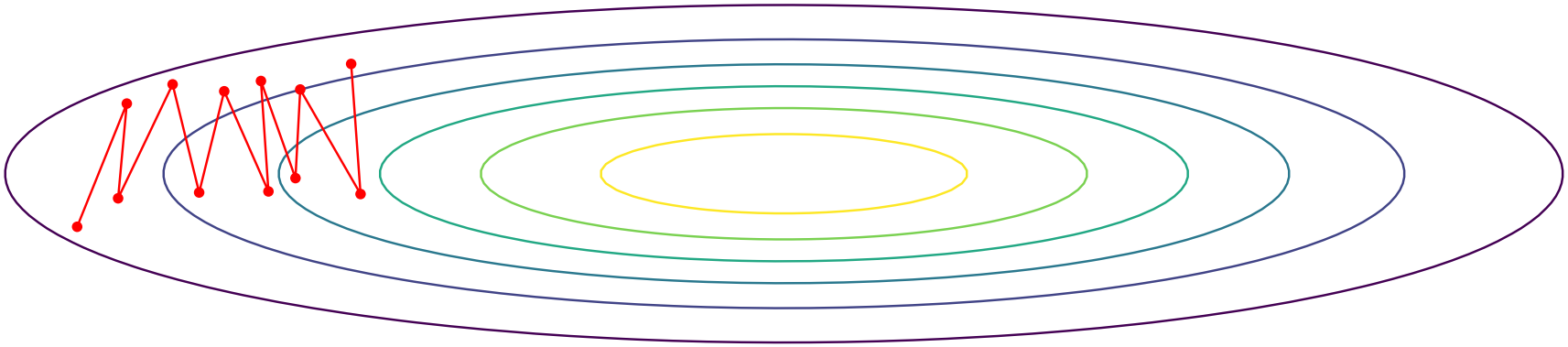
Hyper-parameters: EWMA: ρ; LR: ϵ; with ∇_θ for a mini-batch in iteration i:

- $r_i = \rho r_{i-1} + (1-\rho)\nabla_\theta^2$
- $\theta^* = \theta - \epsilon \frac{\nabla_\theta}{\sqrt{r_i}}$

Stochastic Gradient Descent with RMSprop

Chapter 8.5.2

RMSprop smooths the step size of noisy gradients with EWMA



Hyper-parameters: EWMA: ρ; LR: ϵ; with ∇_θ for a mini-batch in iteration i:

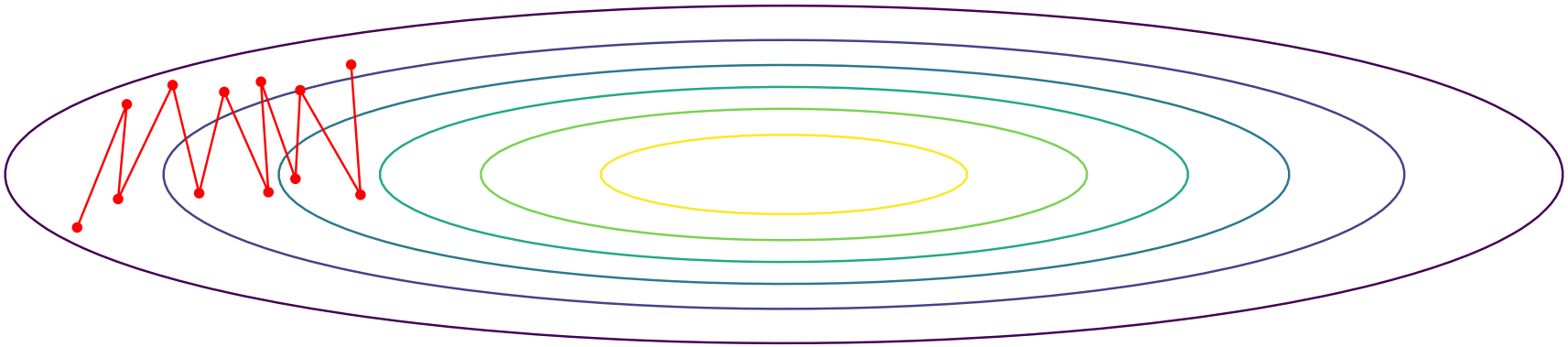
- $r_i = \rho r_{i-1} + (1-\rho)\nabla_\theta^2$
- $\theta^* = \theta - \epsilon \frac{\nabla_\theta}{\sqrt{r_i}}$

To prevent dividing by 0, add a small $\delta \approx 10^{-6}$ to denominator: $r_i = \delta + r_i$
Often implemented without bias correction.
Default setting $\rho = 0.9$

Stochastic Gradient Descent with Adam

Chapter 8.5.3

Adam combines momentum with RMSprop



Hyper-parameters: EWMA: ρ_1, ρ_2; LR: ϵ; with ∇_θ for a mini-batch in iteration i:

- $v_i = \rho_1 v_{i-1} + (1-\rho_1)\nabla_\theta, \qquad r_i = \rho_2 r_{i-1} + (1-\rho_2)\nabla_\theta^2$
- $\hat{v}_i = \frac{v_i}{(1-\rho_1^i)}, \qquad \hat{r}_i = \frac{r_i}{(1-\rho_2^i)}$
- $\theta' = \theta - \epsilon \frac{\hat{v}_i}{\sqrt{\hat{r}_i}}$

To prevent dividing by 0, add a small $\delta \approx 10^{-6}$ to denominator: $r_i = \delta + r_i$
Uses bias correction. Default settings: $\rho_1 = 0.9$ and $\rho_2 = 0.999$,

Questions?

Questions about gradient-statistics based optimization algorithms?

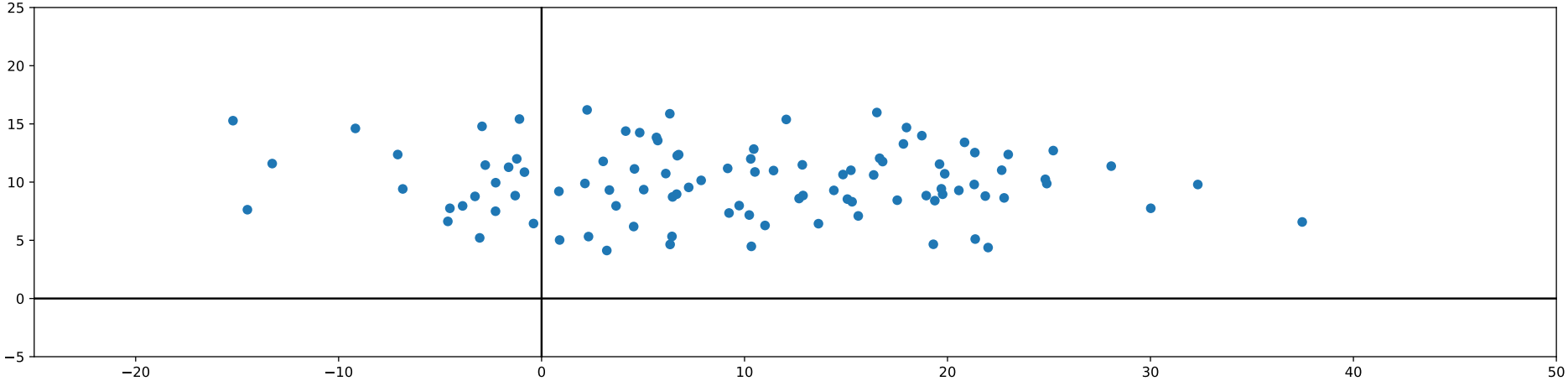
Questions?

Questions about gradient-statistics based optimization algorithms?

New topic: data normalization.

Effect of feature normalization on optimization

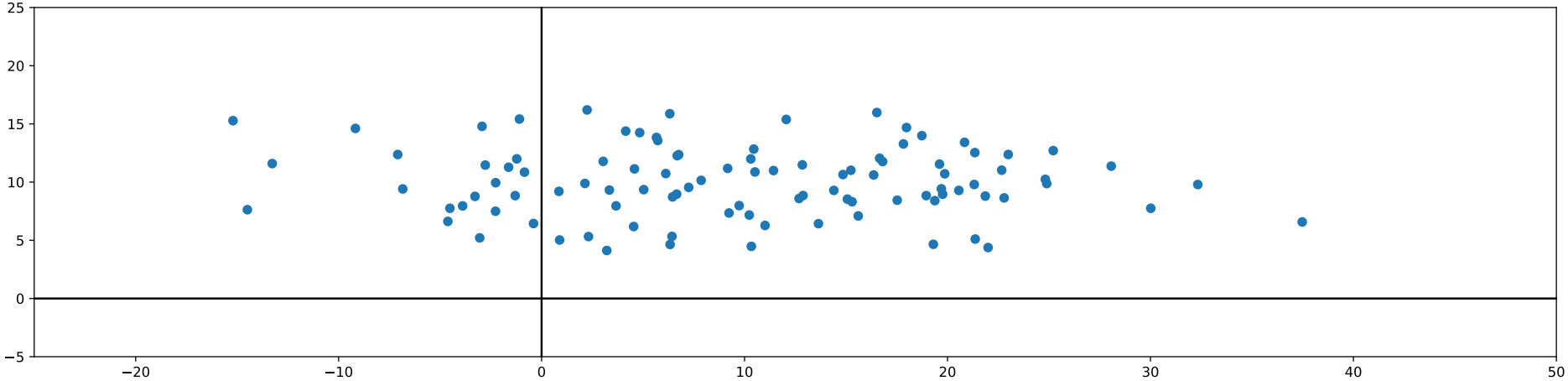
A blue dot is a 2D feature vector (x-axis: feature 1; y-axis: feature 2).



- Centering inputs at zero and equal dimension variance speeds up training

Effect of feature normalization on optimization

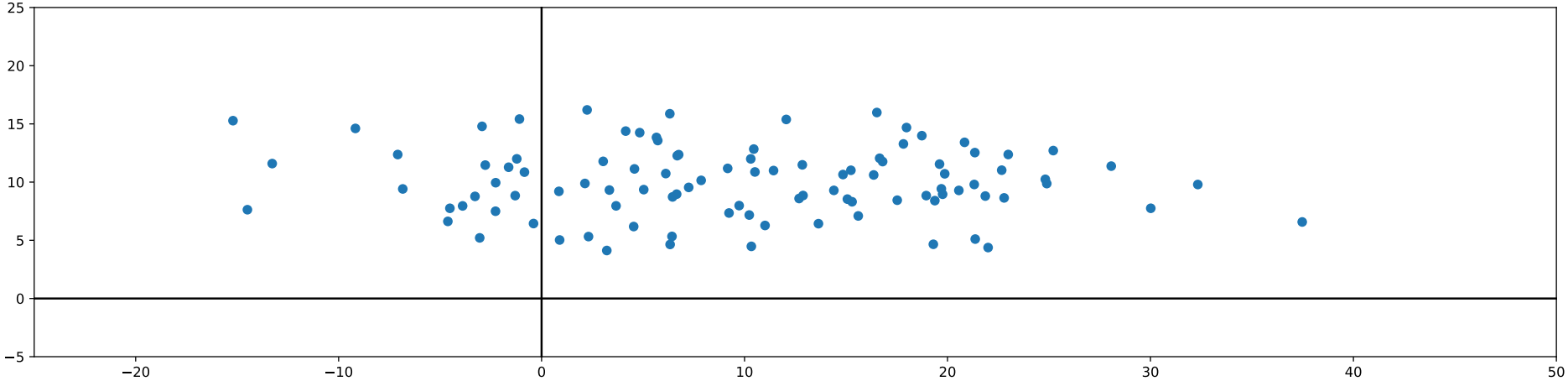
A blue dot is a 2D feature vector (x-axis: feature 1; y-axis: feature 2).



- Centering inputs at zero and equal dimension variance speeds up training
- Q: For data X, how to turn ellipse in a 0-centered circle?

Effect of feature normalization on optimization

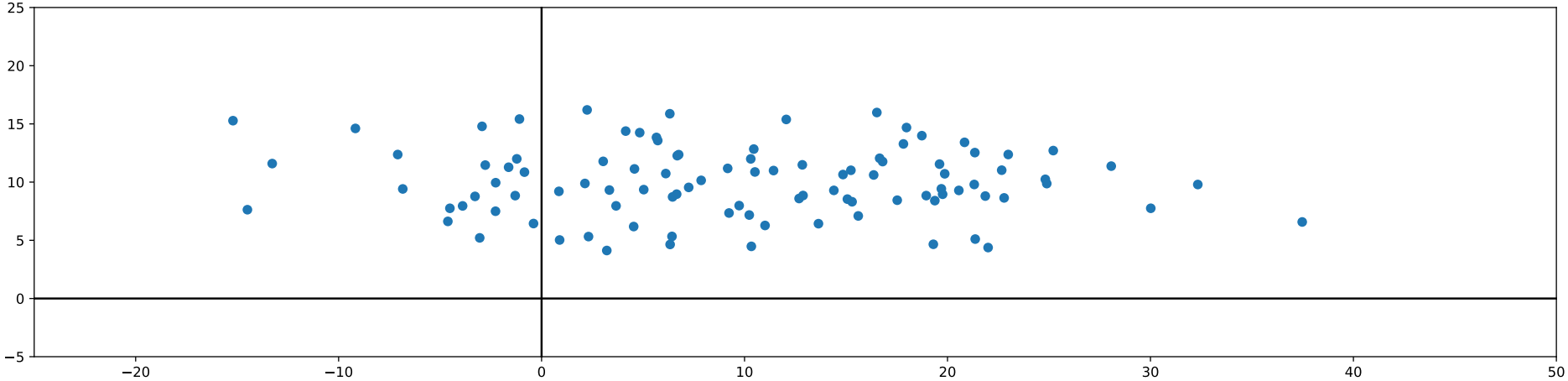
A blue dot is a 2D feature vector (x-axis: feature 1; y-axis: feature 2).



- Centering inputs at zero and equal dimension variance speeds up training
- Q: For data X, how to turn ellipse in a 0-centered circle? A:
- $\mu = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$,

Effect of feature normalization on optimization

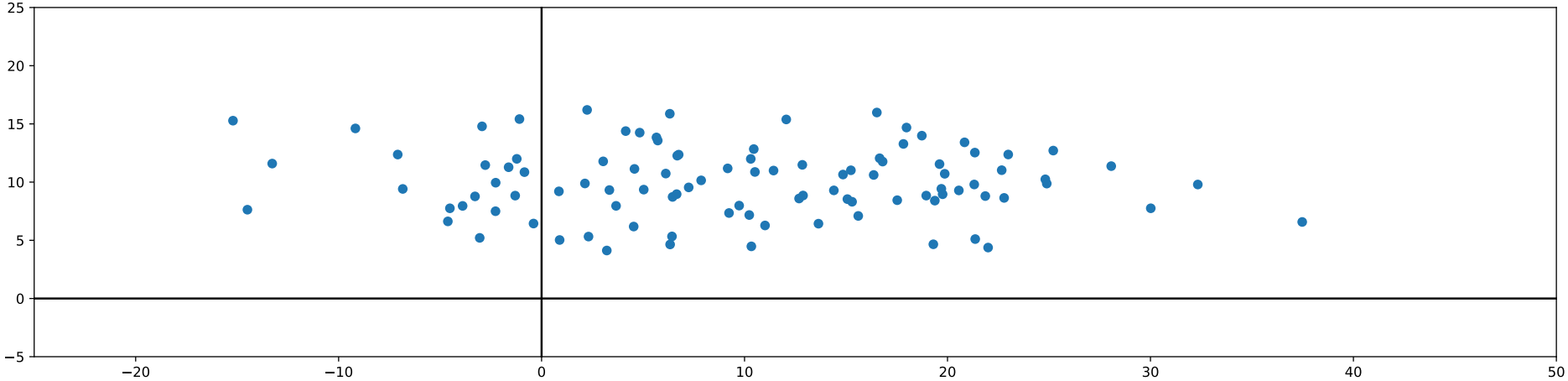
A blue dot is a 2D feature vector (x-axis: feature 1; y-axis: feature 2).



- Centering inputs at zero and equal dimension variance speeds up training
- Q: For data X, how to turn ellipse in a 0-centered circle? A:
- $\mu = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$,
- $\sigma^2 = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)} - \mu)^2$,

Effect of feature normalization on optimization

A blue dot is a 2D feature vector (x-axis: feature 1; y-axis: feature 2).

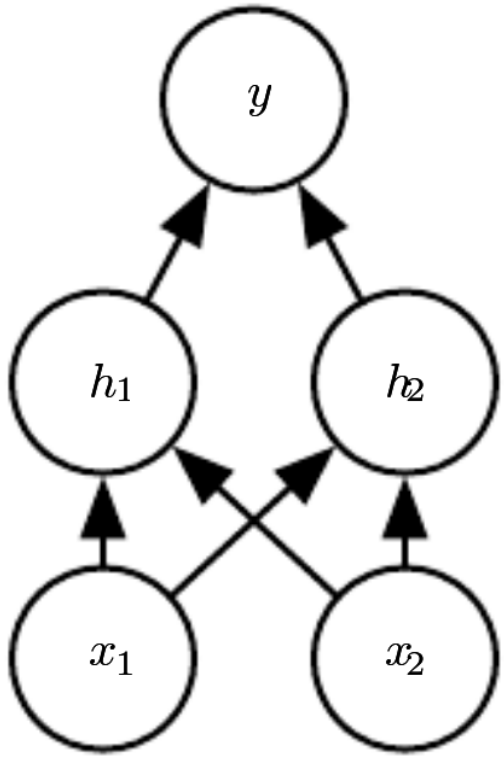


- Centering inputs at zero and equal dimension variance speeds up training
- Q: For data X, how to turn ellipse in a 0-centered circle? A:
- $\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$,
- $\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \mu)^2$,
- $X = \frac{X-\mu}{\sqrt{\sigma^2}}$

Batch norm: Normalizing learned features

Chapter 8.7.1

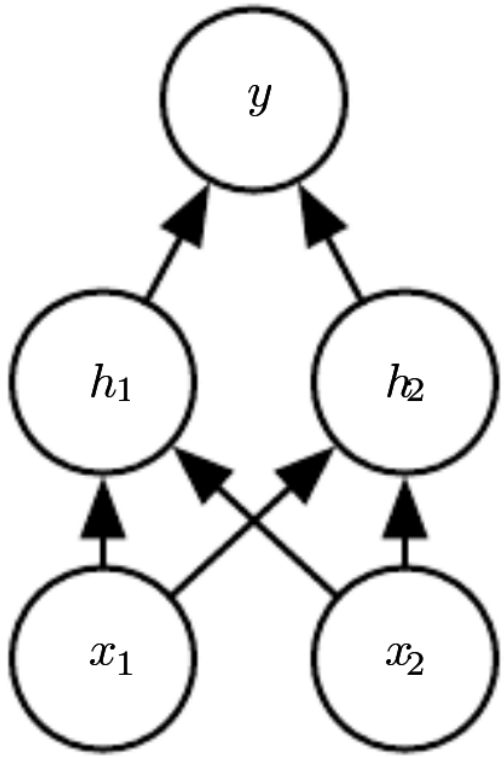
- A deep neural net learns many hidden feature representations.



Batch norm: Normalizing learned features

Chapter 8.7.1

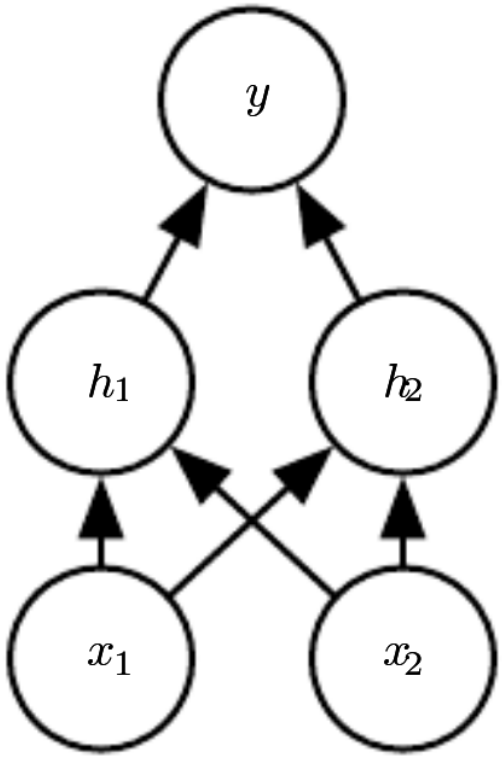
- A deep neural net learns many hidden feature representations.
- Q: How to normalize all those features?



Batch norm: Normalizing learned features

Chapter 8.7.1

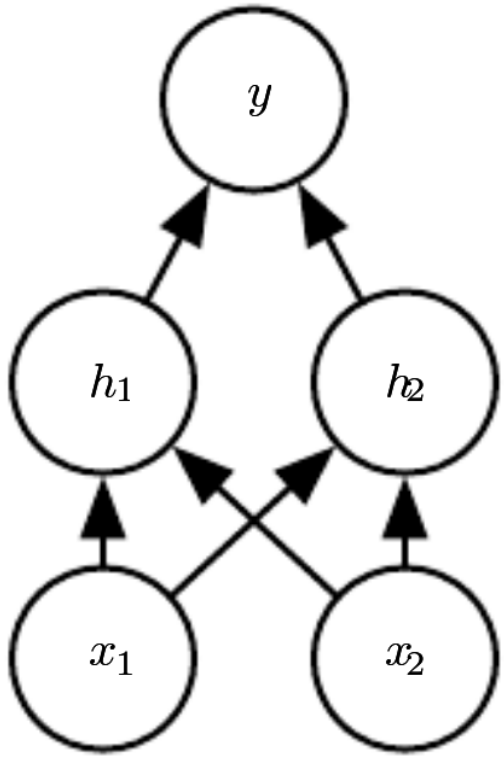
- A deep neural net learns many hidden feature representations.
- Q: How to normalize all those features?
- A: Apply normalization also for each hidden layer.



Batch norm: Normalizing learned features

Chapter 8.7.1

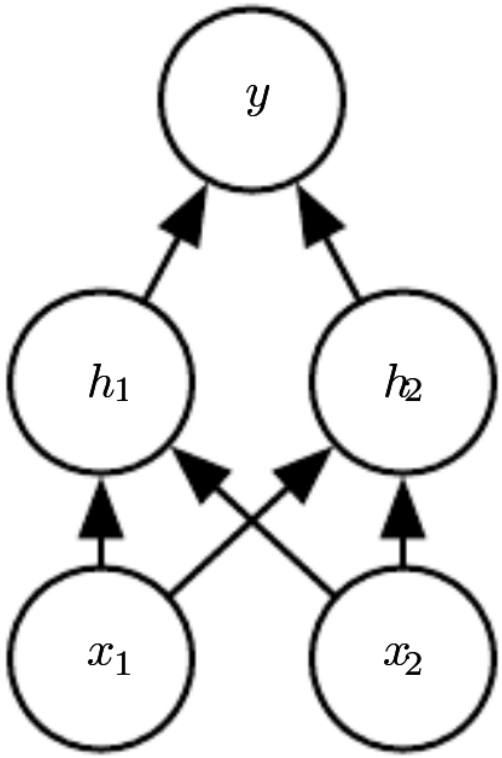
- A deep neural net learns many hidden feature representations.
- Q: How to normalize all those features?
- A: Apply normalization also for each hidden layer.
- Batch norm: Normalize learned features.



Batch norm: Normalizing learned features

Chapter 8.7.1

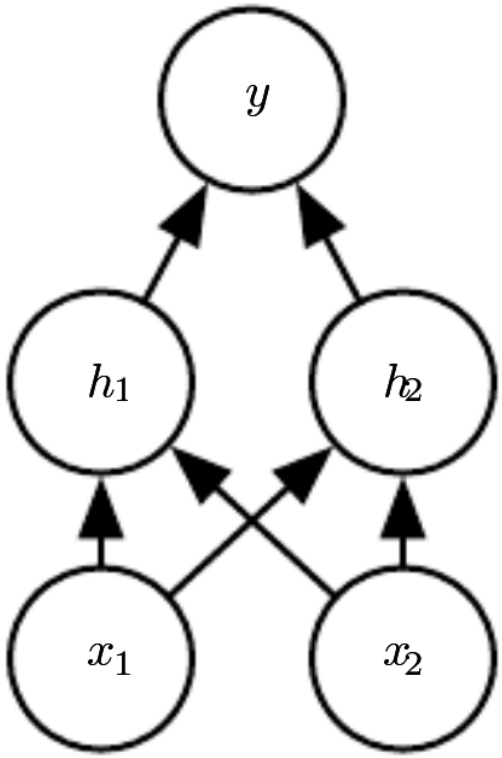
- A deep neural net learns many hidden feature representations.
- Q: How to normalize all those features?
- A: Apply normalization also for each hidden layer.
- Batch norm: Normalize learned features.
- Q: Before or after activation $g(\cdot)$: $h_j = g(z_j) = g(x^\top W_{:,j} + b_j)$?



Batch norm: Normalizing learned features

Chapter 8.7.1

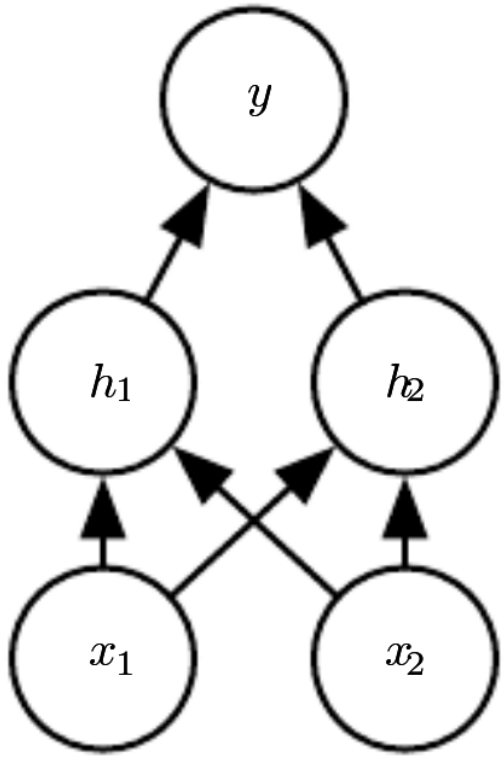
- A deep neural net learns many hidden feature representations.
- Q: How to normalize all those features?
- A: Apply normalization also for each hidden layer.
- Batch norm: Normalize learned features.
- Q: Before or after activation $g(\cdot)$: $h_j = g(z_j) = g(x^\top W_{:,j} + b_j)$?
- A: There is debate; but before.



Batch norm: Normalizing learned features

Chapter 8.7.1, and D2L book[1]

For k samples x_i in mini-batch; for each layer j; $h_j = g(z_i) = g(x_i^\top W_{:,j} + b_j)$:



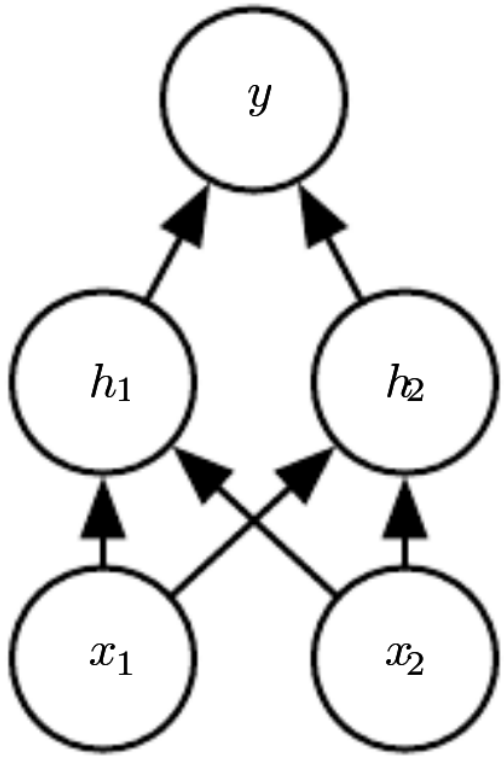
[1] http://d2l.ai/chapter_convolutional-modern/batch-norm.html

Batch norm: Normalizing learned features

Chapter 8.7.1, and D2L book[1]

For k samples x_i in mini-batch; for each layer j; $h_j = g(z_i) = g(x_i^\top W_{:,j} + b_j)$:

- $\mu_i = \frac{1}{k} \sum_{i=1}^{k} z_i$,



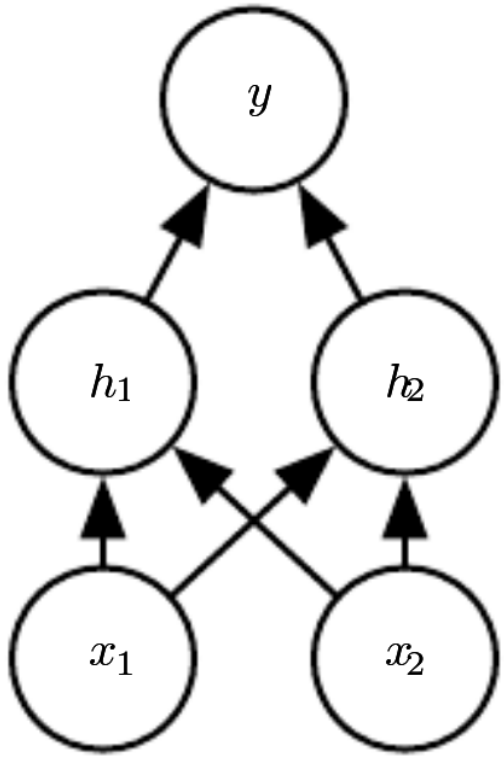
[1] http://d2l.ai/chapter_convolutional-modern/batch-norm.html

Batch norm: Normalizing learned features

Chapter 8.7.1, and D2L book[1]

For k samples x_i in mini-batch; for each layer j; $h_j = g(z_i) = g(x_i^\top W_{:,j} + b_j)$:

- $\mu_i = \frac{1}{k} \sum_{i=1}^{k} z_i$,
- $\sigma_i^2 = \frac{1}{k} \sum_{i=1}^{k} (z_i - \mu_i)^2$,



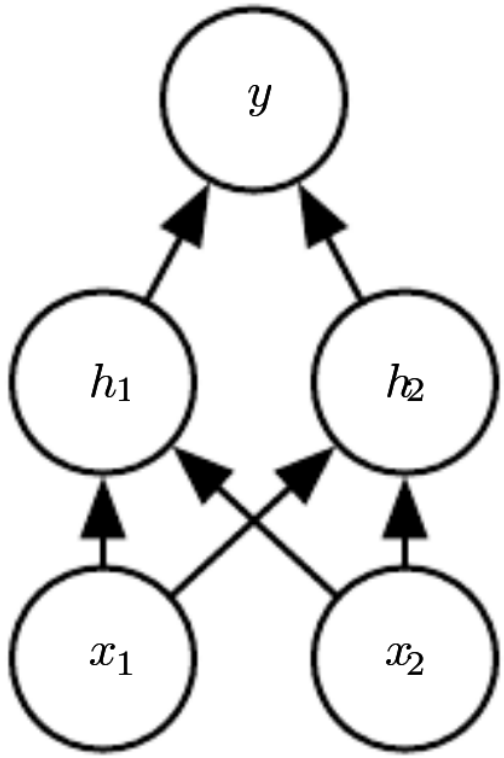
[1] http://d2l.ai/chapter_convolutional-modern/batch-norm.html

Batch norm: Normalizing learned features

Chapter 8.7.1, and D2L book[1]

For k samples x_i in mini-batch; for each layer j; $h_j = g(z_i) = g(x_i^\top W_{:,j} + b_j)$:

- $\mu_i = \frac{1}{k}\sum_{i=1}^{k} z_i$,
- $\sigma_i^2 = \frac{1}{k}\sum_{i=1}^{k}(z_i - \mu_i)^2$,
- $z_i^{\text{norm}} = \frac{z-\mu}{\sqrt{\delta+\sigma^2}}$, (where $\delta \approx 10^{-8}$ prevent div0),



[1] http://d2l.ai/chapter_convolutional-modern/batch-norm.html

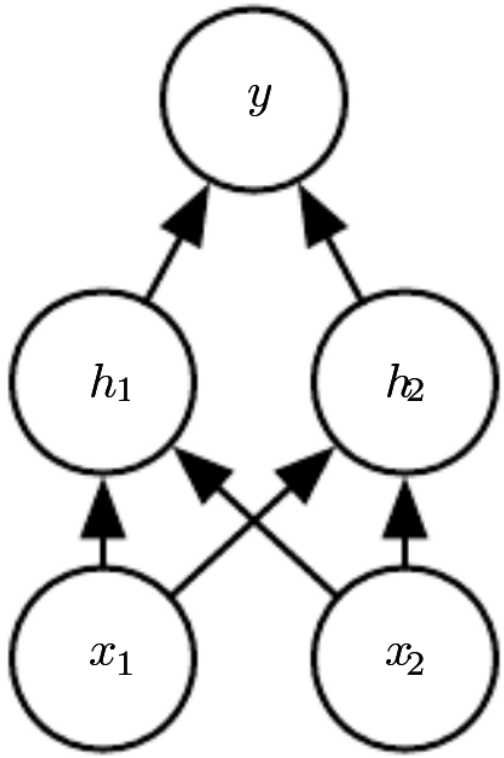
Batch norm: Normalizing learned features

Chapter 8.7.1, and D2L book[1]

For k samples x_i in mini-batch; for each layer j; $h_j = g(z_i) = g(x_i^\top W_{:,j} + b_j)$:

- $\mu_i = \frac{1}{k}\sum_{i=1}^{k} z_i$,
- $\sigma_i^2 = \frac{1}{k}\sum_{i=1}^{k}(z_i - \mu_i)^2$,
- $z_i^{\text{norm}} = \frac{z-\mu}{\sqrt{\delta+\sigma^2}}$, (where $\delta \approx 10^{-8}$ prevent div0),

Sometimes too rigid.



[1] http://d2l.ai/chapter_convolutional-modern/batch-norm.html

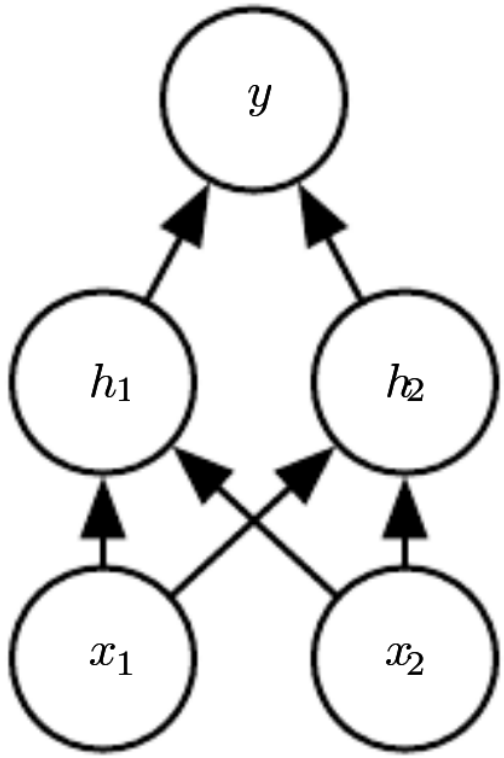
Batch norm: Normalizing learned features

Chapter 8.7.1, and D2L book[1]

For k samples x_i in mini-batch; for each layer j; $h_j = g(z_i) = g(x_i^\top W_{:,j} + b_j)$:

- $\mu_i = \frac{1}{k}\sum_{i=1}^{k} z_i$,
- $\sigma_i^2 = \frac{1}{k}\sum_{i=1}^{k}(z_i - \mu_i)^2$,
- $z_i^{\text{norm}} = \frac{z-\mu}{\sqrt{\delta+\sigma^2}}$, (where $\delta \approx 10^{-8}$ prevent div0),

Sometimes too rigid.
Add learnable parameters γ and β:



[1] http://d2l.ai/chapter_convolutional-modern/batch-norm.html

Batch norm: Normalizing learned features

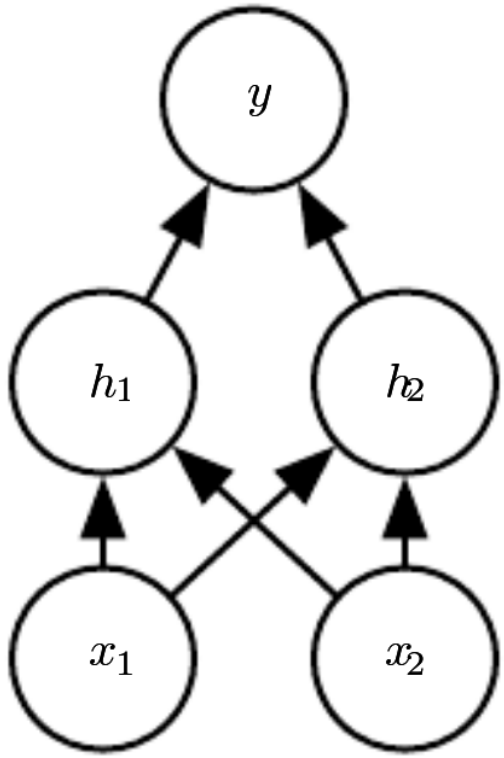
Chapter 8.7.1, and D2L book[1]

For k samples x_i in mini-batch; for each layer j; $h_j = g(z_i) = g(x_i^\top W_{:,j} + b_j)$:

- $\mu_i = \frac{1}{k}\sum_{i=1}^{k} z_i$,
- $\sigma_i^2 = \frac{1}{k}\sum_{i=1}^{k}(z_i - \mu_i)^2$,
- $z_i^{\text{norm}} = \frac{z-\mu}{\sqrt{\delta+\sigma^2}}$, (where $\delta \approx 10^{-8}$ prevent div0),

Sometimes too rigid.
Add learnable parameters γ and β:

- $\overline{z}_i^{\text{norm}} = \gamma_i z_i^{\text{norm}} + \beta_i$



[1] http://d2l.ai/chapter_convolutional-modern/batch-norm.html

Batch norm: Normalizing learned features

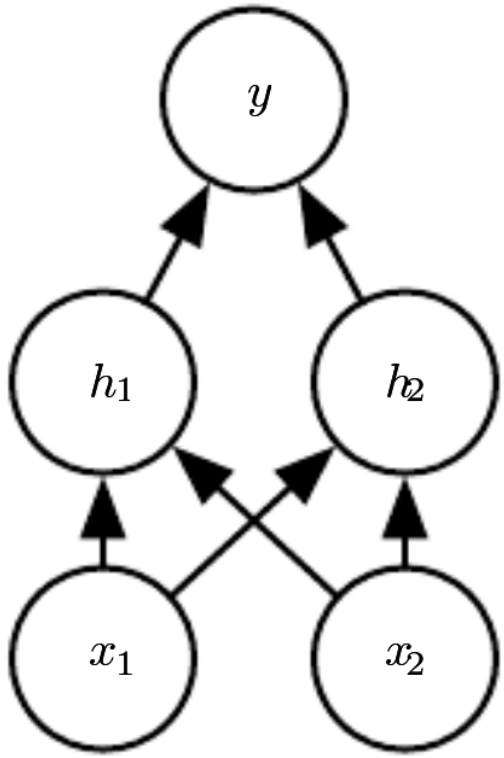
Chapter 8.7.1, and D2L book[1]

For k samples x_i in mini-batch; for each layer j; $h_j = g(z_i) = g(x_i^\top W_{:,j} + b_j)$:

- $\mu_i = \frac{1}{k}\sum_{i=1}^{k} z_i$,
- $\sigma_i^2 = \frac{1}{k}\sum_{i=1}^{k}(z_i - \mu_i)^2$,
- $z_i^{\text{norm}} = \frac{z-\mu}{\sqrt{\delta+\sigma^2}}$, (where $\delta \approx 10^{-8}$ prevent div0),

Sometimes too rigid.
Add learnable parameters γ and β:

- $\overline{z}_i^{\text{norm}} = \gamma_i z_i^{\text{norm}} + \beta_i$
- Network can learn to undo it all:



[1] http://d2l.ai/chapter_convolutional-modern/batch-norm.html

Batch norm: Normalizing learned features

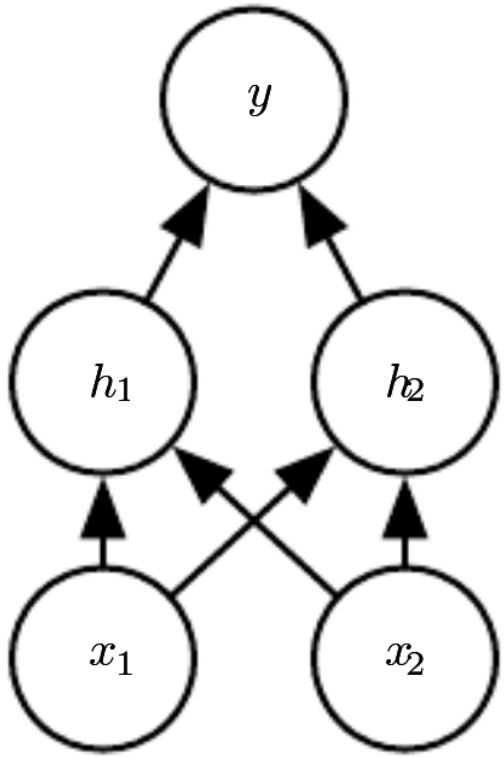
Chapter 8.7.1, and D2L book[1]

For k samples x_i in mini-batch; for each layer j; $h_j = g(z_i) = g(x_i^\top W_{:,j} + b_j)$:

- $\mu_i = \frac{1}{k}\sum_{i=1}^{k} z_i$,
- $\sigma_i^2 = \frac{1}{k}\sum_{i=1}^{k}(z_i - \mu_i)^2$,
- $z_i^{\text{norm}} = \frac{z-\mu}{\sqrt{\delta+\sigma^2}}$, (where $\delta \approx 10^{-8}$ prevent div0),

Sometimes too rigid.
Add learnable parameters γ and β:

- $\overline{z}_i^{\text{norm}} = \gamma_i z_i^{\text{norm}} + \beta_i$
- Network can learn to undo it all:
  Q: How?



[1] http://d2l.ai/chapter_convolutional-modern/batch-norm.html

Batch norm: Normalizing learned features

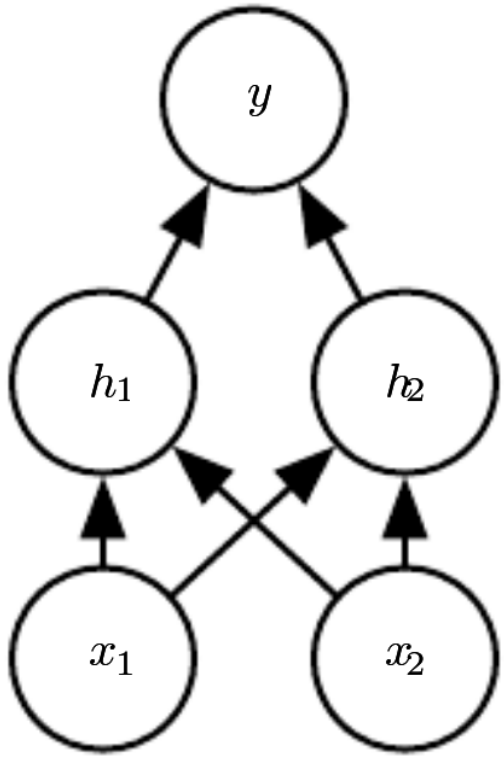
Chapter 8.7.1, and D2L book[1]

For k samples x_i in mini-batch; for each layer j; $h_j = g(z_i) = g(x_i^\top W_{:,j} + b_j)$:

- $\mu_i = \frac{1}{k}\sum_{i=1}^{k} z_i$,
- $\sigma_i^2 = \frac{1}{k}\sum_{i=1}^{k}(z_i - \mu_i)^2$,
- $z_i^{\text{norm}} = \frac{z-\mu}{\sqrt{\delta+\sigma^2}}$, (where $\delta \approx 10^{-8}$ prevent div0),

Sometimes too rigid.
Add learnable parameters γ and β:

- $\overline{z}_i^{\text{norm}} = \gamma_i z_i^{\text{norm}} + \beta_i$
- Network can learn to undo it all:
  Q: How? A: $\gamma_i = \sqrt{\delta + \sigma^2}$ and $\beta_i = \mu$



[1] http://d2l.ai/chapter_convolutional-modern/batch-norm.html

Mini-batches and Batch norm at test time

Chapter 8.7.1,

- Batch norm is applied for every mini-batch.
- It computes μ and σ^2 for each batch.

Mini-batches and Batch norm at test time

Chapter 8.7.1,

- Batch norm is applied for every mini-batch.
- It computes μ and σ^2 for each batch.
- How then to apply it at test time?

Mini-batches and Batch norm at test time

Chapter 8.7.1,

- Batch norm is applied for every mini-batch.
- It computes μ and σ^2 for each batch.
- How then to apply it at test time?
- Use a weighted average of μ and σ^2 of the last mini-batches.

Mini-batches and Batch norm at test time

Chapter 8.7.1,

- Batch norm is applied for every mini-batch.
- It computes μ and σ^2 for each batch.
- How then to apply it at test time?
- Use a weighted average of μ and σ^2 of the last mini-batches.
- How?

Mini-batches and Batch norm at test time

Chapter 8.7.1,

- Batch norm is applied for every mini-batch.
- It computes μ and σ^2 for each batch.
- How then to apply it at test time?
- Use a weighted average of μ and σ^2 of the last mini-batches.
- How? Our friend: the exponential weighted moving average

Mini-batches and Batch norm at test time

Chapter 8.7.1,

- Batch norm is applied for every mini-batch.
- It computes μ and σ^2 for each batch.
- How then to apply it at test time?
- Use a weighted average of μ and σ^2 of the last mini-batches.
- How? Our friend: the exponential weighted moving average
- Use the weighted average for μ and σ^2 and the learned parameters γ and β to compute $\overline{z}_i^{\text{norm}} = \gamma_i z_i^{\text{norm}} + \beta_i$ for each test sample.

Questions?

Recap

- Stochastic Gradient Decent
- Learning rate
- Average and variance of past gradient update statistics help
- Exponentially weighted moving average
- Momentum: Average; RMSprop: variance; Adam: both.
- Feature and Batch normalization